2020

Learning-based perception and control with adaptive stress testing for safe autonomous air mobility

Xuxi Yang
Iowa State University

Learning-based perception and control with adaptive stress testing for safe autonomous air mobility

by

Xuxi Yang

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Aerospace Engineering

Program of Study Committee:
Peng Wei, Co-major Professor
Leifur Thor Leifsson, Co-major Professor
A Ram (Bella) Kim
Jin Tian
Sourabh Bhattacharya

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this dissertation. The Graduate College will ensure this dissertation is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2020

DEDICATION

I would like to dedicate this dissertation to my wife Rui Zhang, without whose support I would not have been able to complete this work.

I would also like to thank my friends and family for their love and support during this work.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects during my Ph.D. study.

This dissertation is the result of a very close collaboration with my advisor, Prof. Peng Wei. Peng has been an amazing mentor throughout my Ph.D. student career. I have learned many great things from him including the way he solves research problems, how he interacts with colleagues and collaborators, and how he communicates research findings through papers and presentations. I have also benefited from his genuine enthusiasm for our research projects and his ever willingness to make time to meet with me.

I would also like to thank Professor A Ram (Bella) Kim, Professor Jin Tian, Professor Leifur Thor Leifsson and Professor Sourabh Bhattacharya for serving on my dissertation and preliminary exam committee. They have provided me a number of valuable suggestions on my dissertation. Their exceptional expertise in consensus dynamics and multi-agent systems has enriched my knowledge in these areas.

I have been very lucky to be a part of Peng's research group, and have the opportunity to interact with our group of talented researchers, Marc Brittain, Josh Bertram, Syed Shihab, Priyank Pradeep, Guodong Zhu, Xufang Zheng, Daniel Zhou, Mohammad Anwar Manjanoor, Abdullah Alnaqeb, Imke Kleinbekman. I am very thankful for the many intellectually inspiring interactions as well as the many friendships that have developed.

I would like to thank my collaborators, for the inspiring intellectual interactions and discussions, their hard work and our great times together, my groupmate Marc Brittain, Josh Bertram and Syed Shihab, Professor Mark Hansen and his Ph.D. student Yulin Liu from the University of California Berkeley, Lisen Deng and Professor Jia Liu from Iowa State University, Professor Husheng Li from the University of Tennessee, Professor Yongming Liu and his Ph.D. student Jueming Hu from

Arizona State University, Professor Nanpeng Yu and his Ph.D. student Jie Shi from the University of California Riverside, Professor Jun Chen and his Ph.D. student Pengcheng Wu from San Diego State University, Professor Chenglong Li and his students from the Civil Aviation Flight University of China, Maxim Egorov and Antony Evans from Airbus UTM, and Michael Murphy from Airmap.

I have greatly benefited from my internships at Acubed by Airbus and Airmap. I would like to thank my colleagues for their guidance and support during my internships, Maxim Egorov, Antony Evans, Karthik Balakrishnan, Richard Golding, Joe Polastre, and Jing Wang from Airbus UTM, Michael Murphy, Adolfo Martinelli, Jacques Joubert, and Gilbert Mena from Airmap.

There are a number of close friends who are not directly related to my research, but their company made this experience so much more enjoyable. I am particularly grateful to Jim Wang and Caixin Yang, who were my roommates when I was doing my internship in Los Angeles, for the time we've been cooking, exercising, and hanging out together. I am also particularly grateful to Daniel Zhou, who offered me a room to sleep during my time in the Bay Area. Linkai Li, who helped me a lot during my first year in the United States. Jiale Feng, who has been a great friend in my last year at Iowa State. Guang Yang, Xiaosong Du, Guodong Zhu, and Xufang Zheng, for the many entertaining lunches and dinners over the past couple of years.

To my wife, Rui Zhang, thanks for flying from China to the United States to spend months with me. Your loving, caring and fun companions are my most valuable memories during my Ph.D. study.

Last and foremost, I am thanking my parents, Zhentong Yang and Xiangzhi Zhang, who have given me unconditional love, support, and encouragement throughout my life.

ABSTRACT

The use of electrical vertical takeoff and landing (eVTOL) aircraft to provide efficient, high-speed, on-demand air transportation within a metropolitan area is a topic of increasing interest, which is expected to bring fundamental changes to the city infrastructures and daily commutes. NASA, Uber, and Airbus have been exploring this exciting concept of Urban Air Mobility (UAM), which has the potential to provide meaningful door-to-door trip time savings compared with automobiles. However, successfully bringing such vehicles and airspace operations to fruition will require introducing orders-of-magnitude more aircraft to a given airspace volume, and the ability to manage many of these eVTOL aircraft safely in a congested urban area presents a challenge unprecedented in air traffic management. Although there are existing solutions for communication technology, onboard computing capability, and sensor technology, the computation guidance algorithm to enable safe, efficient, and scalable flight operations for dense self-organizing air traffic still remains an open question. In order to enable safe and efficient autonomous on-demand free flight operations in this UAM concept, a suite of tools in learning-based perception and control systems with stress testing for safe autonomous air mobility is proposed in this dissertation.

First, a key component for the safe autonomous operation of unmanned aircraft is an effective onboard perception system, which will support sense-and-avoid functions. For example, in a package delivery mission, or an emergency landing event, pedestrian detection could help unmanned aircraft with safe landing zone identification. In this dissertation, we developed a deep-learning-based onboard computer vision algorithm on unmanned aircraft for pedestrian detection and tracking. In contrast with existing research with ground-level pedestrian detection, the developed algorithm achieves highly accurate multiple pedestrian detection from a bird-eye view, when both the pedestrians and the aircraft platform are moving.

Second, for the aircraft guidance, a message-based decentralized computational guidance algorithm with separation assurance capability for single aircraft case and multiple cooperative aircraft case is designed and analyzed in this dissertation. The algorithm proposed in this work is to formulate this problem as a Markov Decision Process (MDP) and solve it using an online algorithm Monte Carlo Tree Search (MCTS). For the multiple cooperative aircraft case, a novel coordination strategy is introduced by using the logit level-k model in behavioral game theory. To achieve higher scalability, we introduce the airspace sector concept into the UAM environment by dividing the airspace into sectors, so that each aircraft only needs to coordinate with aircraft in the same sector. At each decision step, all of the aircraft will run the proposed computational guidance algorithm onboard, which can guide all the aircraft to their respective destinations while avoiding potential conflicts among them. In addition, to make the proposed algorithm more practical, we also consider the communication constraints and communication loss among the aircraft by modifying our computational guidance algorithms given certain communication constraints (time, bandwidth, and communication loss) and designing air-to-air and air-to-ground communication frameworks to facilitate the computational guidance algorithm.

To demonstrate the performance of the proposed computational guidance algorithm, a free-flight airspace simulator that incorporates environment uncertainty is built in an OpenAI Gym environment. Numerical experiment results over several case studies including the roundabout test problem show that the proposed computational guidance algorithm has promising performance even with the high-density air traffic case.

Third, to ensure the developed autonomous systems meet the high safety standards of aviation, we propose a novel, simulation driven approach for validation that can automatically discover the failure modes of a decision-making system, and optimize the parameters that configure the system to improve its safety performance. Using simulation, we demonstrate that the proposed validation algorithm is able to discover failure modes in the system that would be challenging for humans to find and fix, and we show how the algorithm can learn from these failure modes to improve the performance of the decision-making system under test.

CHAPTER 1. INTRODUCTION

1.1 Research Motivation

The increase in road traffic congestion in urban areas is a major concern for commuters and a burden on both the environment and the economy. Every day, millions of person-hours are spent unproductively in cities and billions of gallons of extra fuel burn caused due to the road-traffic congestion [Pradeep and Wei (2019)]. This leads to an interest in Urban Air Mobility (UAM), where the electric vertical take-off and landing (eVTOL) aircraft is able to alleviate transportation congestion by utilizing 3D airspace efficiently for passenger transport in personal commute or on-demand air taxi. A growing community of interest is forming for the concept of UAM including NASA, Uber, Airbus, Honeywell, and many other entities around the globe [Gipson (2017); Thipphavong et al. (2018); Moore (2017); Holden and Goel (2016); Airbus UTM (2017)]. Over a dozen companies including Airbus, Bell, Embraer, Volocopter and Aurora Flight Sciences are building and testing their eVTOL aircraft to make it a reality. The UAM operations are expected to fundamentally change cities and people's lives to reduce commute time and stress. To realize door-to-door time savings for trips, passengers would fly on eVTOL aircraft that can be summoned at any time (i.e. "on-demand"), depart from a local take-off and landing area (TOLA), and land at a TOLA close to their final destination.

However, successfully bringing the UAM operations to fruition will require introducing orders-of-magnitude more aircraft to a given airspace volume [Mueller et al. (2017)], and it is estimated there will be 23,000 aircraft flying major routes within the UAM network by 2035 [Porsche Consulting study (2018)]. Thus the technical challenge is to provide verified scalable systems that enable safe and efficient flight operations for the large number of eVTOL aircraft in the given airspace [Mueller et al. (2017)].

To accommodate the large-scale high-density UAM traffic, traditional Air Traffic Control (ATC) is not suitable. Although the Federal Aviation Administration's (FAA) Next Generation Air Transportation System (NextGen) program aims to modernize ATC and increase the capacity of the airspace [Federal Aviation Administration (2016)], the projected capacity increases over the 20-year period are expected to be no more than 50% [Gawdiak et al. (2009); Timar et al. (2013)], which is sufficient for the increased demand of commercial aircraft but far below the requirement for the UAM air transportation system. In UAM, research efforts to increase the airspace capacity typically rely on aircraft being sufficiently equipped and automated that they can operate relatively independently from the existing ATC system and are therefore not subject to its capacity limits [Moore and Goodrich (2013); Gawdiak et al. (2012)]. Thus in this dissertation, we consider decentralized learning-based perception and control for the aircraft in the UAM, where the proposed decision-making system will run onboard of each aircraft.

In UAM, researchers have proposed structured airspace where the eVTOL aircraft will fly according to fixed routes [Zhu and Wei (2019)]. In this dissertation, we consider a free flight airspace framework [Force (1995)] since it was shown in previous work [Hoekstra et al. (2002); Bilimoria et al. (2003); Consiglio et al. (2007); Blom and Bakker (2015)] that free flight with airborne separation is able to handle a higher air traffic density even in the presence of various uncertainties and delays. In addition, free flight can also increase fuel and time efficiency [Clari et al. (2001)]. In a free flight framework, it is implied that aircraft will be responsible for self-separation assurance and conflict resolution [David J. et al. (2001); Battiste et al. (2002); Barhydt et al. (2005)]. Removing the airway structure may make the process of detecting and resolving conflicts between aircraft more complex. However, previous studies [Tomlin et al. (1998)] show that free flight is potentially feasible because of enabling technologies such as Global Positioning Systems (GPS), data link communications such as Automatic Dependence Surveillance-Broadcast (ADSB) [Kahne and Frolow (1996)], the Next-Generation Airborne Collision Avoidance System (ACAS) [Kochenderfer et al. (2012)], and powerful onboard computation.

In this dissertation, we propose a safe and verified learning-based decentralized perception and decision-making system for eVTOLs to operate in high-density air traffic conditions, through combining the power of onboard aircraft intelligence (vehicle technology) and the advantage of the free flight idea (airspace operation concept), that could enable safe, efficient, and scalable flight operations in UAM.

1.2 Research Objectives

The research objectives of this dissertation are to understand the current state-of-the-art to the following research problems and conduct new research to address these research problems.

1. **Research Problem 1:** with the assumption of shared intention information from cooperative aircraft, or predictable trajectory information from non-cooperative aircraft, how do we design an efficient onboard computational guidance algorithm for one aircraft to help it reach its destination while maintaining safe separation with other aircraft?

2. **Research Problem 2:** for multiple cooperative aircraft, how do we design a decentralized onboard computational guidance algorithm to control these aircraft simultaneously?

3. **Research Problem 3:** in the multi-aircraft case, how do we consider and formulate communication constraints between aircraft and the centralized controller to make the proposed algorithm more practical in real-world application?

4. **Research Problem 4:** assuming the unmanned aircraft are equipped with a camera, how do we enable learning-based perception to detect and track many pedestrians and cars from the aircraft in order to avoid them during a landing event?

5. **Research Problem 5:** given a developed decision-making system as required in Research Problems 1, 2 and 3, how do we perform the verification and validation to ensure such system meets the safety requirements?

1.3 Contributions

In this dissertation, a learning-based perception and control system with adaptive stress testing is proposed for safe autonomous air mobility.

For the aircraft perception, a learning-based onboard computer vision algorithm for detection and tracking is proposed. Previous research works propose traditional computer vision algorithms that run onboard aircraft to detect road, aircraft, moving targets [Frew et al. (2004); Molloy et al. (2017); Saripalli (2009)]. For target tracking, the R-RANSAC multiple moving target tracker [Niedfeldt and Beard (2014, 2015); Niedfeldt et al. (2017); Sakamaki et al. (2017); Ingersoll et al. (2015)] is specifically well suited to track moving targets from a rapidly moving camera, and has excellent track continuity. However, these traditional computer vision algorithms face scalability challenges to be applied in the UAM environment, where there are multiple moving targets with variable shapes (pedestrians and cars) in dynamic complex environments. Thus we propose a learning-based perception algorithm that is able to detect and track multiple moving targets in the urban area through training with large-scale datasets. The learning-based perception system is expected to be faster (more computationally efficient) and more scalable (able to track many moving targets with variable shapes) during a landing event. In addition, we expect this framework can be extended to aircraft detection during en route flight, which is helpful for aircraft separation assurance systems and sense-and-avoid systems.

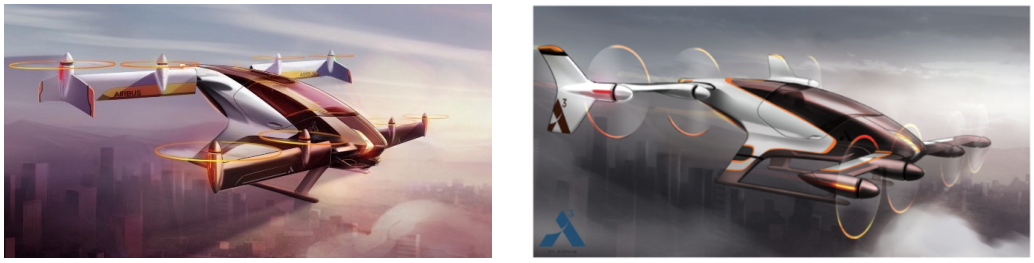
For the aircraft guidance and control, a decentralized, reacting, and cooperative computational guidance algorithm with separation assurance is proposed using Multiagent Markov Decision Process (MMDP) and Monte Carlo Tree Search (MCTS) algorithm. Previous research works mainly solve this problem in an off-line manner using optimal control based on centralized or decentralized methods. Centralized algorithms typically require long computation time, which is hard to scale to multiple aircraft in the UAM environment. Decentralized approaches are often shortsighted and can not look ahead more than one step. MDP based methods can take account the long term or downstream effect of the sequential decisions, but solving the formulated MDP is usually time-consuming. In this dissertation, we propose a fast and efficient MDP and MCTS based algorithm

that takes advantage of decentralized control and online MDP solutions, which is shown to scale to multiple cooperative aircraft in real time. Besides, we also consider the communication constraints among the aircraft to make the proposed algorithm more practical, by modifying the proposed computational guidance algorithms given certain communication constraints (time, bandwidth, and communication loss) and designing air-to-air and air-to-ground communication frameworks to facilitate the proposed computational guidance algorithm.

Although the proposed decision-making algorithms can scale to a much larger number of multiple aircraft, system validation and safety verification are necessary to ensure the developed autonomous systems meet the high safety standards of aviation. Verification for decision-making systems can be categorized into three groups: formal methods based online verification, formal methods based offline verification, and simulation-driven stress testing. Online verification or safety guards are implemented as a "wrapper" of the learning-based controllers. It checks the decisions provided by the controller in real time and resorts to a safe option when identifies an unsafe decision [Mihatsch and Neuneier (2002); Even-Dar et al. (2006); Lötjens et al. (2019); Raju et al. (2019)]. Offline formal methods attempt to prove the safety properties of a decision-making system by constructing a mathematical model of the system based on reachability analysis [Van Wesel and Goodloe (2017); Liu et al. (2019); Jeannin et al. (2015); Huang et al. (2017); Katz et al. (2017); Ashok et al. (2018)], but in general does not scale to systems with multiple agents and large, continuous state/action space. Simulation-driven verification approaches use a dynamic simulation model to evaluate the performance of a system using a finite number of simulation paths or scenarios, which is more flexible and computational tractable for offline verification. As a more efficient way of finding most-likely failure scenarios, adaptive stress testing has been recently proposed as a practical approach [Lee et al. (2019, 2015); Koren et al. (2018)]. The key idea of adaptive stress testing is a carefully designed reward function that leads to the discovery of the system's failure modes. However, these approaches can be computationally expensive because they require either learning or computing a stress testing policy. In this dissertation, we build upon the adaptive stress testing paradigm, but propose a new sample efficient methodology for finding the failure modes of

UTM decision-making systems using Bayesian optimization method that does not involve learning or pre-computing, and expand the proposed approach in a way that enables the methodology to also improve the decision-making system based on the identified failure modes.

To demonstrate the proposed decision-making system, a high-density free flight airspace simulator in the OpenAI Gym environment is built, where the aircraft dynamics are modeled based on the tandem tilt-wing eVTOL (Airbus Vahana) from Airbus A^3 [Pradeep and Wei (2018a)] which has flown over 80 full-scale test flights [Airbus (2018)]. Figure 1.2 shows the take-off and landing phase and cruise phase of Vanaha aircraft. In this dissertation, we restrict our scope to the cruise phase of this aircraft in en route airspace. For the scheduling and spacing services in the vertiport terminal airspace (arrival and departure management), readers can refer to [Pradeep and Wei (2018b); Kleinbekman et al. (2018); Bertram and Wei (2020b)]. Note this open-source simulator is very flexible, which is able to model structured and non-structured airspace, aircraft in en route flight phase, static obstacles (such as buildings), and communication constraints.



(a) take-off and landing phase (b) cruise phase

Figure 1.2: Airbus Vahana with tandem tilt-wing configuration during the take-off phase and cruise phase [Stoschek (2017)].

1.4 Organization of the Dissertation

This dissertation is organized as follows: in chapter 2, related literature related to separation assurance and conflict resolution is reviewed, which helps us achieve the first research objective. In chapter 3, the computational guidance algorithm for the single aircraft case is proposed. In chapter 4, we consider the computational guidance algorithm for the multiple cooperative aircraft case. In

chapter 5, the communication framework is proposed for the algorithm to make it more practical. In chapter 6, the learning-based perception algorithm is presented. In chapter 7, we propose the sample efficient validation framework used to validate the autonomous systems. In chapter 8, the contributions and future works are pointed out.

CHAPTER 2. LITERATURE REVIEW AND BACKGROUND

2.1 Conflict Resolution and Collision Avoidance Systems

Decades of research have explored a variety of approaches for designing collision avoidance systems for both manned and unmanned aircraft, large commercial aircraft and small unmanned aircraft. For conflict detection and resolution of commercial aircraft, there exist several surveys: [Kuchar and Yang (2000)] gives a comprehensive survey of air traffic conflict detection and resolution systems. [Netjasov and Janic (2008)] provides a high-level outline in safety risk analysis, and a recent survey [Mitici and Blom (2018)] presents a unified mathematical framework for air traffic conflict and collision definitions and methods to estimate the probability of conflict and collision.

Currently, the Traffic Alert and Collision Avoidance System (TCAS) is the only widely-deployed aircraft collision avoidance system, which is required on all large transport aircraft in the world. If the system predicts that the intruder will penetrate a predefined safety buffer, the system will issue a resolution advisory to the pilot to adjust the vertical speed of the aircraft [SC-186 (2006)]. Recent work on formulating the problem of collision avoidance as a Partially Observable Markov Decision Process (POMDP) has led to the development of the ACAS X family of collision avoidance systems [Kochenderfer et al. (2012); Kochenderfer and Chryssanthacopoulos (2011)]. The version for manned aircraft, ACAS Xa, is expected to become the next international standard for large commercial transport and cargo aircraft. Both TCAS and ACAS are designed to resolve one on one conflicts between aircraft with vertical maneuvers. The difference between them is TCAS uses fixed rules to resolve the conflicts while ACAS uses a probabilistic model (solving a Markov Decision Process (MDP) problem with discrete state space), which leads to a better performance than TCAS. For multiple aircraft case, the Autonomous Operations Planner (AOP) developed by NASA [Karr et al. (2012)] is a flexible and powerful prototype of a flight-deck automation system to support self-separation of aircraft while en route. It incorporates a variety of algorithms that

provide flight crew support for strategic and tactical conflict resolutions and conflict-free trajectory planning while meeting route constraints and avoiding airspace hazards. In this dissertation, we investigate how to resolve conflicts for multiple aircraft and guide the aircraft to their destinations through a series of actions by formulating this problem as a MDP and solving it using Monte Carlo Tree Search (MCTS) algorithm.

For UAS Traffic Management, NASA's UTM project is aiming at enabling the increasing number of low-altitude, small UAS operations in uncontrolled airspace [Kopardekar et al. (2016)], specifically to enable safe and efficient en route UAS operations for civilian and public applications. The Integrated Configurable Algorithms for Reliable Operations of Unmanned Systems software architecture (ICAROUS), being developed as part of the UTM project, will provide highly assured core software modules for building safety-centric autonomous unmanned aircraft applications [Consiglio et al. (2016)]. A recent implementation known as DAIDALUS (Detect and Avoid Alerting Logic for Unmanned Systems) [Muñoz et al. (2015)] is the basis for the ICAROUS software architecture. The core logic of DAIDALUS consists of: (1) definition of self-separation threshold (SST) and well-clear violation volume, (2) algorithms for determining if there exists a potential conflict between aircraft pairs within a given lookahead time, and (3) a determine-processing functionality that provides maneuver guidance and alerting logic.

While research on a UTM system [Kopardekar et al. (2016)] for small UAS (sUAS) operating at low altitudes is relevant for UAM, it provides services appropriate for small UAS that do not always easily extend to the UAM environment [Mueller et al. (2017)]. For example, the risk of human injuries in the collision of two sUAS is very low [Jung et al. (2016)], while larger vehicles with humans onboard will present significantly higher risks and the safety standards will have to be significantly enhanced for eVTOL aircraft in UAM over sUAS in UTM. Also, sUAS have the freedom to take off and land nearly anywhere, while eVTOL aircraft in UAM will be restricted to a network of vertiports and therefore require scheduling and spacing services in vertiport terminal airspace.

In this dissertation, we mainly focus on guidance and conflict resolution systems for these new entrants (such as eVTOL urban air mobility aircraft) in the metropolitan airspace. In the applications for these new entrants, the existing work can be categorized based on the following criteria. We will briefly discuss the related work categorized based on the first criterion: centralized method and decentralized method. The second and third criteria will be also discussed.

- Centralized/Decentralized [Schouwenaars et al. (2004)]: whether the problem is solved by a central supervising controller (centralized) or by each aircraft individually (decentralized).
- Planning/Reacting [Siegwart et al. (2011)]: the planning approach generates feasible or even optimal paths ahead of execution; whereas the reacting approach typically uses an onboard collision avoidance system to respond to obstacles and other vehicles.
- Cooperative/Non-cooperative: whether there exists communication among aircraft, or between aircraft and the central controller.

2.1.1 Centralized Algorithms

In centralized methods, the conflicts between aircraft are resolved by a central supervising controller. Most of them are planning ahead of time. Under such scenarios, the state of each aircraft, the obstacle information, and the trajectory constraint (such as required times of arrival and restricted airspace area) are known to the central controller (thus centralized methods are always cooperative), and the central controller in return designs the whole trajectory for each aircraft pre-departure or en route, typically by formulating it to an optimal control problem. These methods can be based on semidefinite programming [Frazzoli et al. (2001)], nonlinear programming [Raghunathan et al. (2004); Enright and Conway (1992)], mixed integer linear programming [Schouwenaars et al. (2001); Richards and How (2002); Pallottino et al. (2002); Vela et al. (2009)], mixed integer quadratic programming [Mellinger et al. (2012)], sequential convex programming [Augugliaro et al. (2012); Morgan et al. (2014)], evolutionary techniques [Delahaye et al. (2010); Cobano et al. (2011)], and particle swarm optimization [Pontani and Conway (2010)]. Besides for-

mulating this problem using optimal control framework, computational geometry methods such as visibility graph [Hoffmann et al. (2004)] and Voronoi diagrams [Howlet et al. (2004)] can also handle the path planning problem for aircraft without modeling detailed vehicle dynamics. However, calculating the exact solution for such computational geometry based robot motion planning will become intractable [Canny (1988)] when the state space becomes large or high-dimensional. To address this issue, sample-based planning algorithms are proposed, such as probabilistic roadmaps [Kavraki et al. (1994)], RRT [LaValle (1998)], and RRT* [Karaman and Frazzoli (2011); Čáp et al. (2013)]. These centralized methods often generate the whole trajectories for agents. However, as the number of aircraft grows (in multi-agent case), the computation time of these methods typically scales exponentially. Moreover, these centralized planning approaches typically need to be re-run, as new information in the environment is updated (e.g., a new aircraft enters the airspace, or one aircraft failed to execute its planned trajectory).

2.1.2 Decentralized Algorithms

On the other hand, decentralized methods scale better with respect to the number of agents and are more robust since they are not vulnerable to a single point of failure. However, since the agents act only on local information, global optimality of a decentralized control policy is often hard to achieve [Pallottino et al. (2006)]. In decentralized methods, all the conflicts are resolved by each aircraft individually. Decentralized methods can be cooperative and non-cooperative. Researchers have proposed several algorithms under the case where the communication between aircraft can be successfully established (cooperative with communication) [Wollkind et al. (2004)]. Algorithms in [Purwin et al. (2008); Desaraju and How (2011)] are based on message-passing schemes, which resolve local (e.g., pairwise) conflicts without needing to form a joint optimization problem between all members of the team. In [Schouwenaars et al. (2004)], every agent is allotted a time slot to compute a dynamically feasible and collision-free path using mixed integer linear programming. In [Inalhan et al. (2002)], the authors recast the global optimization problem as several local problems, which are then iteratively solved by the agents in a decentralized way. In the Decentralized Model

Predictive Control approach [Richards and How (2004)], the aircraft solve their own sub-problem one by one and send the action to other subsystems through communication.

There are also scenarios where communication cannot be reliably established (non-cooperative) and the aircraft will take action at each time step based on the sensor information. Many works fall in this category: Model Predictive Control [Shim and Sastry (2007); Shim et al. (2003)] can be used to solve the collision avoidance problem but the computation load is relatively high. Potential field method [Khatib and Mampey (1978)] is computationally fast. However, a navigation function [Koditschek and Rimon (1990)] is required to deal with the local minima problem and make it a complete path planner [Rimon and Koditschek (1988); Connolly et al. (1990)], which involves discretizing the state space. With the help of machine learning and reinforcement learning [Kahn et al. (2017); Ong and Kochenderfer (2016); Chen et al. (2017); Li et al. (2019); Brittain and Wei (2019)], collision avoidance algorithm (without trajectory re-planning to the destination) can have a promising performance, but the data collection and model training part are expensive. Using the Monte Carlo Tree Search algorithm to solve this problem [Yang and Wei (2018)] does not need model training and the algorithm can finish in any predefined computation time, but the aircraft can only adopt several discretized actions at each time step. In [Wolf and Kochenderfer (2011)] the authors proposed a sample-based POMDP approximation algorithm that can run onboard for continuous state and observation spaces, which can find the optimal action using the branch and bound method. However, computation time was the most limiting factor in this work. Geometry based algorithms [Han et al. (2009); Park et al. (2008); Krozel et al. (2000); Van Den Berg et al. (2011)] can be also applied for collision avoidance problem and the computation time only grows linearly with the increasing number of aircraft. The drawback of these geometric approaches is that they cannot look ahead for more than one step (they only pay attention to the current action and do not take account of the effect of subsequent actions) and the outcome can be local optimal in the view of the global trajectory.

In this dissertation, under the free flight framework, we propose a computational guidance algorithm with a separation assurance capability, which is a message-passing based decentralized,

reacting, and cooperative algorithm. We formulate this computational guidance problem as a Multiagent MDP (MMDP) and solve the formulated MMDP using the MCTS algorithm that can run onboard the aircraft. In fact, the proposed algorithm in this dissertation can be either centralized (where a centralized controller is responsible for gathering aircraft state information and issuing action advisories to all of the aircraft) or decentralized (where the algorithm runs onboard the aircraft and aircraft can coordinate with each other through wireless communication). We described both centralized and decentralized cases in detail in our recent paper [Yang et al. (2020)]. In this dissertation, we focus on the decentralized case. There are similar works using MDP formulation which solve this problem offline in the pre-departure phase [Temizer et al. (2010); Kochenderfer et al. (2012); Ong and Kochenderfer (2016)] or online during the en route phase [Bertram and Wei (2020a)]. Offline solvers require large computation time up front to compute the optimal policy for the full state space and discrete MDP formulations. Offline methods are typically not adaptive to changes in the environment because the policy is determined ahead of time. Also, the state space of many problems is too large to adequately represent as a finite set of enumerable states. Comparing with offline methods, by using longer onboard computation time, onboard methods address the shortcomings of offline methods by planning only for the current state and a small number of possible plans. Since onboard algorithms only need to plan for the current state that can take any continuous value, state discretization of MDP formulation is not required. Onboard algorithms are also able to account for changes in the environment because they are executed once at each decision point, allowing for updates between these points. There is also POMDP formulation for this problem which aims to consider the state uncertainty due to the sensor noise [Wolf and Kochenderfer (2011)], where the authors use a search algorithm to solve this POMDP. The major difference between this paper and our work is that they use depth-first search and do not consider the state transition uncertainty, while we use the robust MCTS algorithm to handle the dynamical model uncertainty through simulations.

2.1.3 Airspace Sectorization

Airspace sectorization is a concept widely used in commercial aviation. According to the FAA [Federal Aviation Administration (2019)], the airspace sector is defined to be an airspace area with predefined horizontal and vertical dimensions for which a controller or group of controllers has air traffic control responsibility, normally within an air route traffic control center or an approach control facility. Sectors are established based on predominant traffic flows, altitude strata, and controller workload.

Airspace sectors can be created to deal with the high demand for aircraft traffic [NATS (2019)]. For example, in times when there are high levels of air traffic, more sectors may be opened with more controllers allocated to manage the aircraft within an area of airspace. This is done to maintain safety as a controller can only manage a certain number of aircraft at one time. In this dissertation, we use airspace sectors to reduce the computation time for the proposed computational guidance algorithm.

While the airspace sectorization can help reduce the computation time by distributing the workload to several sectors, it also introduces complexity by requiring additional coordination between aircraft and sector controllers, especially when an aircraft flies across more than one airspace sector.

Note that the airspace sectorization also introduces additional "hand-off" conflicts between aircraft near the boundary between (at least) two sectors [SKYbrary (2019b)], since the adjacent sectors might have different plans to resolve the conflict and not able to see the traffic situation in the neighboring sector. This type of loss of separation deserves special attention because of the hand-off between two sectors. Several typical scenarios that could cause hand-off conflicts include poor or missing coordination/communication between sectors, aircraft flying along the sector boundary, and different minimum separation standards used in adjacent sectors. Also, adverse weather avoidance, communication equipment failure, high controller workload, transfer of control too early/too late, and sector skipping can contribute to conflicts between aircraft as well. For more details, readers can refer to [SKYbrary (2019b)].

In this dissertation, we will design a novel mechanism for airspace sectorization to resolve the hand-off conflict mentioned above.

2.2 Object Detection Algorithms for Aircraft Perception

2.2.1 Classical Object Detectors

Before deep-learning techniques were applied to object detection, the sliding-window paradigm was the state-of-the-art. One of such first object detectors is the Viola Jones Object Detector [Viola et al. (2001)], which was primarily used for facial detection. With the introduction of Histogram of Oriented Gradients (HOG) [Dalal and Triggs (2005)], SVM-based classifiers became an effective method for pedestrian detection. Afterwards, the Deformable Part Model(DPM), also based on HOG [Felzenszwalb et al. (2009)], reigned as the state-of-the-art algorithm for many years.

2.2.2 Two-stage Detectors

Another object-detection paradigm is the two-stage detector. The first stage is a region-proposal model that generates a set of candidate proposals from the image. These candidates should have a high probability of being objects and low probability of being background. The second-stage classifier labels these candidates as different categories (either object classes or background).

The R-CNN Model [Girshick et al. (2014)] is known as the first object detector that proposed the two-stage approach. R-CNN applies a non deep learning model, Selective Search [Uijlings et al. (2013)], as the region proposal model and a convolutional neural network (CNN) is applied to the generated candidates independently to output the classes. With the introduction of Region of Interest pooling (also known as RoI pooling), Fast R-CNN [Girshick (2015)] further reduces the computation time of R-CNN by only performing the CNN forward computation on the image as a whole. Faster R-CNN [Ren et al. (2015)] replaces selective search with a region-proposal network (RPN), which reduces the number of proposed regions generated, while ensuring precise object detection. Mask R-CNN [He et al. (2017)] uses the same basic structure as Faster R-CNN,

but adds an RoI alignment layer to help locate objects at the pixel level and further improve the precision of object detection.

With the region-proposal model, two-stage detectors achieve good detection accuracy. However, the inference of two-stage detectors with these region proposals requires prohibitive computation time, making detection slow, and thus loses appeal for real-time detection.

2.2.3 One-Stage Detectors

One-Stage detectors are based on global regression and classification and work directly from image pixels to yield bounding-box coordinates and class probabilities of objects in the image. This can reduce computation time, making them more favorable than two-stage detectors.

YOLO [Redmon et al. (2016)] makes use of the whole topmost feature map to detect the objects, but has difficulty detecting small objects and unusual aspect ratios. To fix this, SSD [Liu et al. (2016)] was then proposed; it has good performance on multi-scale detection. RetinaNet [Lin et al. (2017b)] is a single-step object detector which achieves state-of-the-art accuracy by introducing a novel loss function called Focal Loss to deal with the class imbalance issue. This model represents the first instance where one-stage detectors have surpassed two step detectors in accuracy while retaining superior speed.

In this dissertation, we will use RetinaNet as our algorithm to detect pedestrians and cars from drone video.

2.2.4 Perception Algorithms for Aircraft

Interest in Unmanned Aerial Vehicles (UAVs) has grown tremendously in recent years. Nowadays, more and more powerful and agile UAVs are recruited for civilian applications in terms of surveillance and infrastructure inspection. The major challenge today is the development of autonomously operating aerial agents capable of completing missions independently of human interaction. To this extent, visual sensing techniques have been integrated into the control pipeline of

the UAVs in order to enhance their navigation and guidance skills [Kanellakis and Nikolakopoulos (2017)].

Previous research works propose traditional computer vision algorithms that run onboard aircraft to detect road, aircraft, moving targets [Frew et al. (2004); Molloy et al. (2017); Saripalli (2009)]. For target tracking, the R-RANSAC [Niedfeldt and Beard (2014, 2015); Niedfeldt et al. (2017); Sakamaki et al. (2017); Ingersoll et al. (2015)] is specifically well suited to tracking multiple moving targets from a rapidly moving camera, and has excellent track continuity. However, these traditional computer vision algorithms face scalability challenges to be applied in the UAM environment, where there are multiple moving targets with variable shapes (pedestrians and cars) in dynamic complex environments.

For aircraft in the UAM environment, there are also several other restrictions to deploy the onboard perception algorithms. First, due to the limited memory and computing power of embedded onboard devices, the trained model needs to be pruned to a smaller size for real-time object detection [Zhang et al. (2019); Ringwald et al. (2019)]. Also, the resolution increase in visual sources and relatively small scale of pedestrians makes the problem even harder by raising the expectations to leverage all the details in images [Ozge Unel et al. (2019)]. In this dissertation, we use model pruning and image cropping to deal with the above mentioned challenges.

2.3 Optimization Based Adaptive Stress Testing

The adaptive stress testing methodology relies on simulation to discover failure modes of a system under test. This is done by modifying properties of the environment in which the system operates such that the system fails, or by directly guiding the dynamics of the system into a failure mode. The characteristics of the environment or the dynamic path of the system that led to the failure can then be directly examined and addressed.

Consider a system that is being tested by a domain expert in Figure 2.1. In this testing environment, the domain expert chooses the input parameters for the simulator, performs the simulation, and examines the output metrics of interest after the simulation is completed. Based

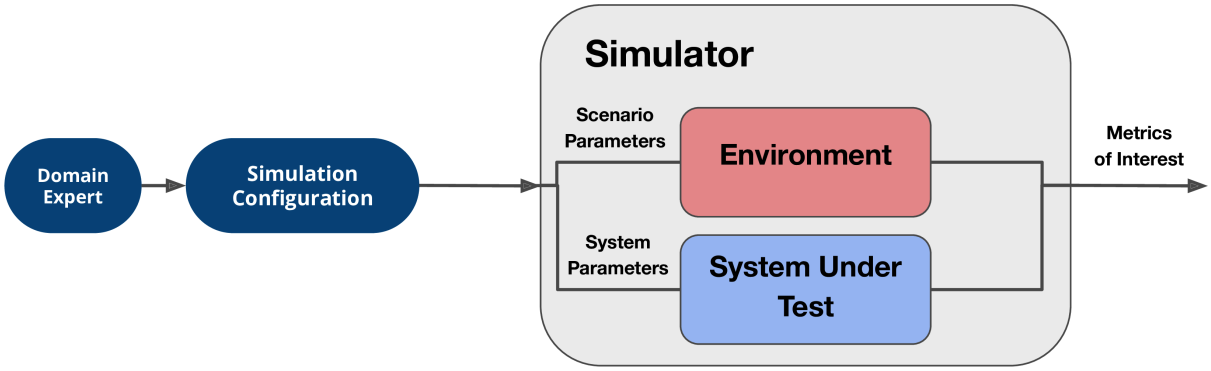


Figure 2.1: Simulation-driven testing of a generic system that can be parameterized and evaluated in a configurable environment.

on the output metrics, the domain expert may choose to modify the simulation parameters to further test the system. The traditional adaptive stress testing methodology, creates a feedback loop between the metrics of interest generated by the simulation and the parameters input into the simulation. Traditionally, the modified parameter is typically the random seed that configures any random numbers generated by the environment [Lee et al. (2015)], and the system under stress is held fixed.

In this work, we make two critical changes to the adaptive stress testing methodology: (1) we close the feedback loop using an efficient black box optimizer instead of an MDP solver, and (2) we allow the system to modify both the environment parameters and the system parameters (see Figure 2.2). These modifications reduce the problem of adaptive stress testing to the task of performing an iterative optimization, where the goal of the optimizer is to discover the simulation scenarios or the simulation paths that lead to failure modes. The objective of the optimization can be constructed by using the output metrics from the simulator that serve as indicators of system failure. For autonomous traffic management systems, metrics tracking near mid air collisions (NMACs) tend to be strong indicators of system failure, and are adopted in this work. The parameter space over which the optimizer is searching is represented by the joint parameter space

of the environment and the system under test. The result of the optimization is a set of environment scenarios and a corresponding system parameterization that leads to the failure of that instance of the system being tested. For example, consider an adaptive test case attempting to validate an autonomous strategic deconfliction service. The objective is formulated using the frequency of NMACs in the simulation, and the optimization discovers that a package delivery scenario with four intersecting operations leaves one to be in conflict with a system parameterization that favors trajectory deviations over delaying the start time of the flight. We apply this methodology in the remainder of the dissertation to discover unique and non-trivial failure modes that cover a variety of UTM use cases.

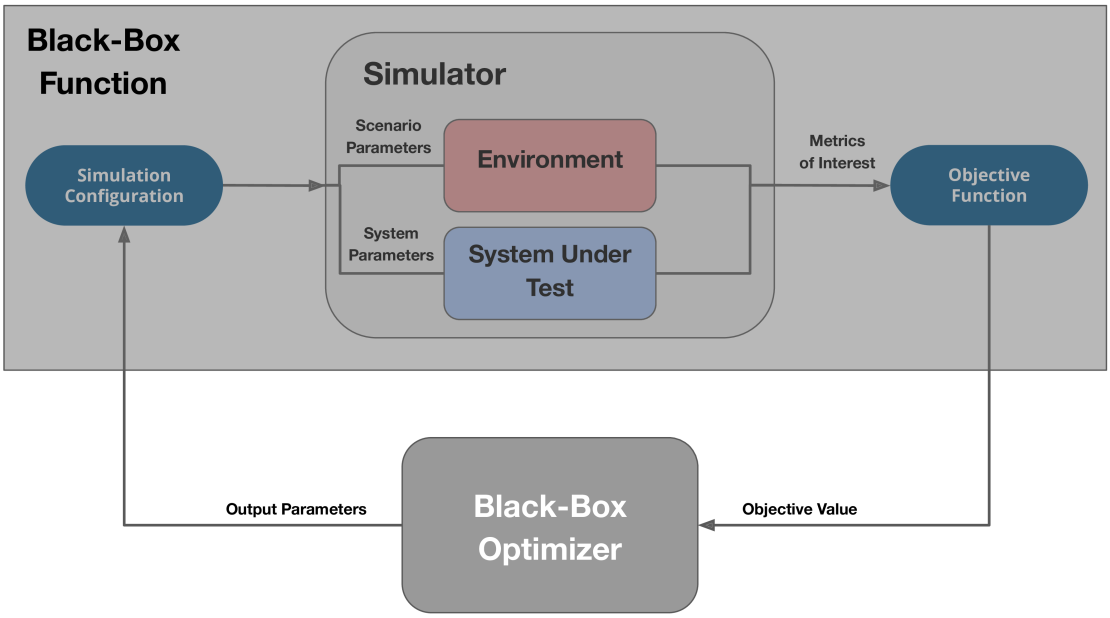


Figure 2.2: Adaptive stress testing performed with a black-box optimizer.

2.4 Background

In this section, we briefly review the background of the Markov Decision Process and Monte Carlo Tree Search, as well as the Multiagent Markov Decision Process.

2.4.1 Markov Decision Process (MDP)

Since the 1950s, MDPs [Bellman (1957)] have been well studied and applied to a wide area of disciplines [Howard (1964); White (1993); Feinberg and Shwartz (2012)], including robotics [Koenig and Simmons (1998); Thrun (2002)], automatic control [Mariton (1990)], economics, and manufacturing [Puterman (2014)]. In a MDP, the agent may choose any action a that is available based on current state s at each time step. The process responds at the next time step by moving into a new state s' with certain transition probability, and gives the agent a corresponding reward r.

More precisely, the Markov Decision Process (MDP) includes the following components:

- The state space $\mathcal{S}$ which consists of all the possible states.
- The action space $\mathcal{A}$ which consists of all the actions that the agent can take.
- Transition function $\mathcal{T}(s_{t+1}|s_t, a_t)$ which describes the probability of arriving at state s_{t+1}, given the current state s_t and action a_t.
- The reward function $\mathcal{R}(s_t, a_t, s_{t+1})$ which decides the immediate reward (or expected immediate reward) received after transitioning from state s to state s', due to action a. In general, the reward will depend on the current state, current action, and the next state. However, the reward function may only depend on the current state s_t, which will be the case in this dissertation.
- A discount factor $\gamma \in [0, 1]$ which decides the preference on immediate reward versus future rewards. Setting the discount factor less than 1 is also beneficial for the convergence of cumulative reward.

In a MDP problem, a policy π is a mapping from the state to one specific action (known as deterministic policy)

$$\pi : \mathcal{S} \rightarrow \mathcal{A} \tag{2.1}$$

The goal of MDP is to find an optimal policy π^* that, if followed from any initial state, maximizes the expected cumulative rewards over all the future steps:

$$\pi^* = \underset{\pi}{\text{argmax}} E[\sum_{t=0}^{T-1} R(s_t, a_t, s_{t+1})|\pi] \tag{2.2}$$

Q-function and value function are two important concepts in MDP. The optimal Q-function $Q^*(s, a)$ means the expected cumulative reward received by an agent starting in state s and picks action a, then will behave optimally afterwards. Therefore, $Q^*(s, a)$ is an indication of how good it is for an agent to pick action a while being in state s. The optimal value function $V^*(s)$ denotes the maximum expected total reward when starting from state s, which can be expressed as the maximum of $Q^*(s, a)$ over all possible actions:

$$V^*(s) = \max_a Q^*(s, a) \quad \forall s \in \mathcal{S} \tag{2.3}$$

2.4.2 Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search (MCTS) is a method for finding optimal decisions in a given domain by taking random samples in the decision space and building a search tree according to the results [Coulom (2006); Browne et al. (2012)]. It has already had a profound impact on Artificial Intelligence (AI) approaches for domains that can be represented as trees of sequential decisions, particularly games and planning problems [Silver et al. (2016, 2017a); Champandard (2014)], including the current state-of-art computer program AlphaZero in the Game of Go [Silver et al. (2017b)].

The basic MCTS process is conceptually easy to understand, where a tree is built in an incremental and asymmetric manner, as shown in Figure 2.3 (from [Browne et al. (2012)]). For each iteration of the algorithm, a tree policy is used to find the most urgent node of the current tree. The tree policy attempts to balance considerations of exploration (look in areas that have not been

well sampled yet) and exploitation (look in areas which appear to be promising). A simulation is then rolled out from the selected node and the search tree updated according to the result. This involves the addition of a child node corresponding to the action taken from the selected node and an update of the statistics of its ancestors. Moves are made during this simulation according to some default policy, which in the simplest case is to make uniformly random moves. A great benefit of MCTS is that the values of intermediate states do not have to be evaluated, as for depth-limited minimax search, which greatly reduces the amount of domain knowledge required. Only the value of the terminal state at the end of each simulation is required.

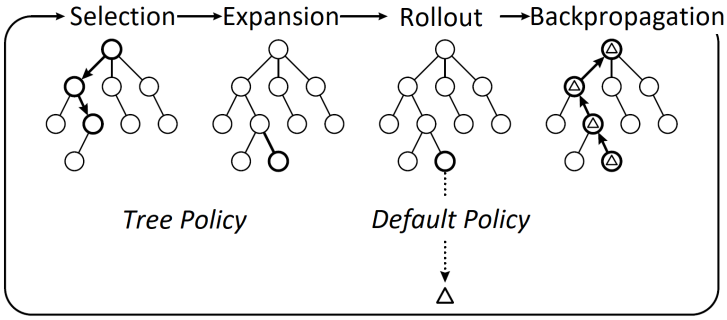


Figure 2.3: One iteration of general MCTS approach [Browne et al. (2012)].

2.4.3 Multiagent Markov Decision Process (MMDP)

Besides applying the proposed computational guidance algorithm to only one single aircraft, in this dissertation we also consider how the MCTS algorithm can scale to multiple cooperative agents.

In order to extend MDPs to multiagent settings, Boutilier [Boutilier (1996); Boutilier et al. (1999); Boutilier (1999)] has introduced Multiagent Markov Decision Processes (MMDPs), which allow for representing sequential decision-making problems in cooperative multiagent settings. Similar to MDP, MMDP is defined as a tuple $< n, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} >$ [Sigaud and Buffet (2013)] where

- n is the number of agents in the whole system.
- $\mathcal{S}$ is the set of states s.

- $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ denotes the set of joint actions, where $\mathcal{A}_i$ is the set of individual actions of agent i.
- $\mathcal{T}$ is the transition function which gives the probability $\mathcal{T}(s_{t+1}|s_t, a_t)$ that the process moves to state s_{t+1} when the agents execute the joint action $a \in \mathcal{A}$ from state s_t.
- $\mathcal{R}(s_t, a_t, s_{t+1})$ is the reward obtained when the process state changes from s_t to s_{t+1} under the influence of joint action a_t.

Although MMDP can model the interactions among multiple agents, there are also many challenges in MMDP [Busoniu et al. (2010)]:

1. The curse of dimensionality will cause exponential growth of the discrete state-action space in the number of state and action variables. For example, assume we have 10 agents, and each agent has 3 actions at each time step, then there will be 3^{10} different action combinations to consider at each time step, and each action could be the optimal one.
2. Nonstationarity arises in multiagent systems because each agent is facing a moving-target learning problem: the best policy changes as the other agents' policies change. For example, as shown in Figure 2.4, without knowing the action of the other aircraft, we cannot tell which action is optimal since each action could lead to a LOS event between them. And even if we get the optimal action for both aircraft (e.g., both take action to turn right), the optimal action of one aircraft will be affected if the other aircraft changed their actions.

Most of the literature studied multiagent systems in stochastic environments with a focus on Nash equilibrium [Nash et al. (1950)] or long-term stable behaviors. But when we are dealing with real-time decision-making systems, information about Nash equilibrium is not always helpful [Kochenderfer (2015)]. First of all, it may be unclear which equilibrium to adopt if there are many different equilibria in the system. For systems with only one equilibrium, it may be difficult to compute the Nash equilibrium when the computation time is limited [Daskalakis et al. (2009)]. An area known as behavioral game theory [Camerer (2011)] aims to model agents that are limited

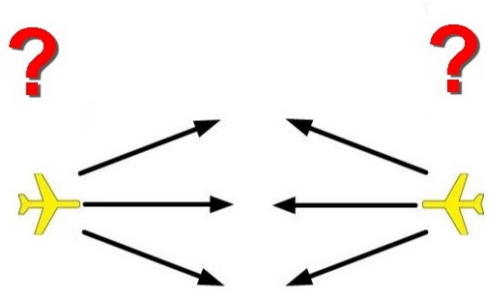


Figure 2.4: The optimal action selected by one aircraft also depends on the action selected by the other aircraft.

in the number of steps of strategic look-ahead when the decision time is limited. Many different behavioral models exist, but the logit level-k model [Stahl II and Wilson (1994); Stahl and Wilson (1995)] has become popular recently and tends to work well in practice. In the logit level-k model, an agent at logit level-k assumes all of the other agents follow logit level-$(k-1)$ strategy. In this way, the actions of other agents become fixed thus the tree search process can avoid the action explosion issue in the multi-agent case.

In this dissertation, we use a variant of the logit level-k model to make decisions for all of the agents in a fast, dynamic and uncertain environment where real-time onboard decisions are needed.

CHAPTER 3. SINGLE-AGENT COMPUTATIONAL GUIDANCE WITH SEPARATION ASSURANCE FOR AUTONOMOUS URBAN AIR MOBILITY OPERATIONS

3.1 Introduction

In this chapter, we start off with one aircraft (single agent) and address the research problem 1 listed in Section 1.2. Research problem 1 can be seen as a subproblem of research problem 2, where we propose the computational guidance algorithm applied for single aircraft, with the assumption of shared intention information from cooperative aircraft, or predictable trajectory information from non-cooperative aircraft. The objective in this chapter is to guide the aircraft to its destination as soon as possible, while maintaining safe separation with any other intruder aircraft with known or predictable near future trajectories.

In this chapter, we first describe this problem mathematically and formulate it as a MDP. Then we present the designed MCTS algorithm to solve the formulated MDP. We also describe how to apply the optimal reciprocal collision avoidance (ORCA) method to solve this problem. Numerical experiment results and conclusions are shown at the end of this chapter.

3.2 Problem Formulation

3.2.1 Problem Statement

The goal of this chapter is to control a single aircraft through a series of actions so that the aircraft can arrive at the destination while avoiding potential conflict with other intruder aircraft during the flight. This is a sequential decision-making problem that can be formulated as a MDP problem. In this MDP problem, the action is decided directly from the state, which incorporates

all the information (the position and velocity of intruder aircraft, the position of the destination) for the agent to decide which action is optimal for the corresponding state.

In this dissertation, a high-density free flight airspace scenario is considered: all the intruders (the aircraft we are not controlling) can only fly straight at a fixed velocity (this assumption can be relaxed if we have the flight plan or flight intention of the intruder aircraft), and only one aircraft (the ownship) is equipped with MCTS algorithm and will try to avoid the conflicts with other intruder aircraft. Here we use the aircraft performance data from Airbus Vahana [Vertical Flight Society (2018)] for the aircraft.

When controlling the aircraft, only horizontal actions are considered in this dissertation, which means all the aircraft will be flying at the same altitude and this problem can be solved in 2 dimensions. This assumption is reasonable because UTM limits its focus to a narrow altitude band between 200 and 500 feet [Federal Aviation Administration (2015)].

Besides, we also assume the aircraft can get the intruder aircraft information (position and velocity) through the sensor perfectly. In future work, we will test the performance of this algorithm under different levels of measurement uncertainties [Allignol et al. (2017); Blom and Bakker (2015)].

The objectives for this specific MDP problem are two-fold: the first is to guide the aircraft to the goal state in a short time, and the second is to avoid any conflicts between the controlled aircraft and other intruder aircraft. Therefore, the reward function should be able to capture both two objectives.

Based on the above description, this problem will be mathematically formulated as a MDP problem in the next subsection.

3.2.2 MDP Formulation

3.2.2.1 State Space:

A state includes all the information the ownship needs for its decision making: the position and velocity of all the aircraft including ownship and intruders, together with the goal position. For the intruders, we use (x, y), (v_x, v_y) to denote its position and velocity. For the ownship, besides its

current position (x, y) and velocity (v_x, v_y), the speed v, the heading angle ψ, and the bank angle ϕ are also included in the state, we can control the aircraft by changing its acceleration and bank angle at each time step. To sum up, if there are n intruders, 1 ownship, and 1 goal, it will need $4 \times n + 7 \times 1 + 2$ values to describe the current state.

Note here the state space is continuous (e.g., all the variables of a state can take continuous values). In general, for a MDP with continuous state variables, it is not clear how to best represent the policy, since it is impossible to enumerate all possible state-action mappings. For previous MDP-based algorithms to solve conflict avoidance problems, some possible approaches to represent the policy include using a grid-based discretization of the state space $\mathcal{S}$ and the action space $\mathcal{A}$ [Kochenderfer and Chryssanthacopoulos (2011); Ong and Kochenderfer (2016)] or using policy compression techniques [Julian et al. (2016, 2018)]. The advantage of the MCTS algorithm is that it does not need to discretize the state space. For each state, the MCTS algorithm will generate action onboard for the aircraft to follow in real-time.

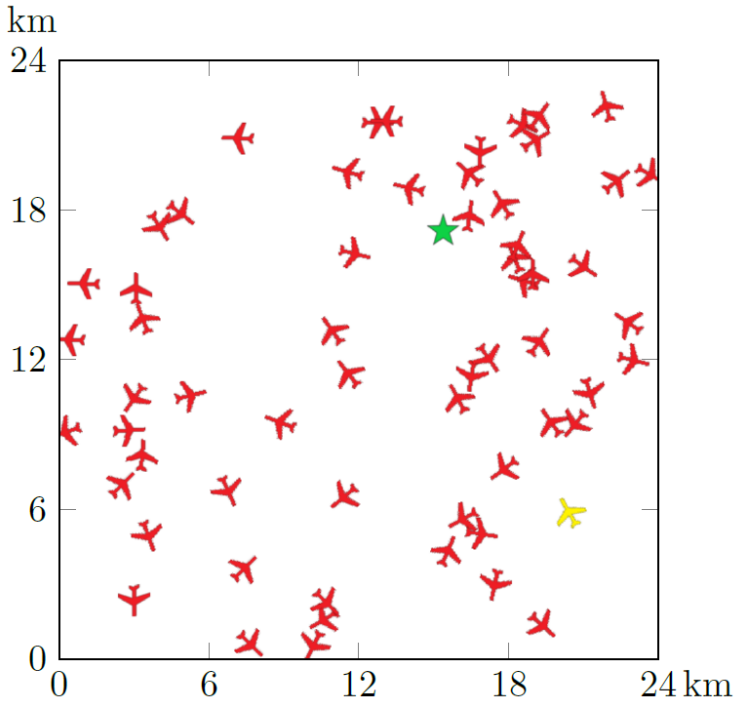


Figure 3.1: An example state of the MDP formulation.

Figure 3.1 shows an example state of this MDP. This state includes all the aircraft information in a $24km \times 24km$ map. It should be noted that the displayed aircraft size in this figure is not proportional to its size in the real world, which is approximately 6m by 6m [Vertical Flight Society (2018)]. In this figure, the yellow aircraft is the ownship, the red aircraft are the intruder, and the green star is the goal position for the ownship. At current state, we denote the position and velocity of k-th intruder aircraft as $(i_x^{(k)}, i_y^{(k)})$ and $(i_{vx}^{(k)}, i_{vy}^{(k)})$. For the ownship, we denote its position and velocity as (o_x, o_y) and (o_{vx}, o_{vy}), its speed as o_v, its heading angle as o_ψ, and its bank angle as o_ϕ, and we use (g_x, g_y) to describe the goal position for the ownship. So if there are n intruder aircraft, one ownship, one goal position in this airspace, then the state will be $(i_x^{(1)}, i_y^{(1)}, i_{vx}^{(1)}, i_{vy}^{(1)}, \cdots, i_x^{(n)}, i_y^{(n)}, i_{vx}^{(n)}, i_{vy}^{(n)}, o_x, o_y, o_{vx}, o_{vy}, o_v, o_\psi, o_\phi, g_x, g_y)$. This state will include sufficient information for the aircraft to make optimal decision at each time step.

3.2.2.2 Action Space

At the beginning of each time step (5 sec), the ownship can choose to change both its bank angle and acceleration at certain rates.

For the bank angle, the ownship can choose to turn right, turn left, or go straight. More precisely, the advisory for the change of bank angle constitutes the action set

$$\mathcal{A}_\phi = \{-5^\circ/s, 0^\circ/s, +5^\circ/s\} \tag{3.1}$$

where negative corresponds to the right turn and positive to left turn. For the passenger comfort, we restrict the bank angle to lie between -25° and 25°. When the heading angle action leads the bank angle to go beyond $\pm 25^\circ$, the bank angle will be clipped to $\pm 25^\circ$.

For the acceleration, the ownship can choose one acceleration from the action set

$$\mathcal{A}_a = \{-5m/s^2, 0m/s^2, 5m/s^2\} \tag{3.2}$$

Similar to the bank angle, we restrict the speed to be in between $50m/s$ and $80m/s$ [Pradeep and Wei (2018a)] following the aircraft performance data of Airbus Vahana [Airbus (2016); Lovering (2016); Vertical Flight Society (2018)].

At each time step, the ownship will choose one action $(a_\phi, a_a) \in \mathcal{A}_\phi \times \mathcal{A}_a$ and the ownship will maintain the action during this time step.

It's natural to consider extending the set of actions (conflict resolution advisories) to include more options than 9. However, using the MCTS algorithm to calculate the optimal action will be more time consuming with the extended action space since the tree size will grow exponentially with the number of actions. Because computation time is an important factor for the online algorithm, some techniques can be used for extending action space in future steps, such as truncated Monte Carlo search algorithms [Tesauro and Galperin (1997)] or using a policy network to narrow down the search to high-value actions [Silver et al. (2016)]. In this dissertation, we use 3 by 3 action space to keep our scope more focused.

3.2.2.3 Dynamic Model

Based on the current state and current action, the following kinematic model will be used to compute state transition for ownship:

$$\begin{aligned} \dot{v} &= a_a \\ \dot{\phi} &= a_\phi \\ \dot{\psi} &= \frac{g \tan \phi}{v} \\ \dot{x} &= v \cos \psi \\ \dot{y} &= v \sin \psi \end{aligned} \tag{3.3}$$

where a_a is the acceleration, a_ϕ is the changing rate of bank angle, ϕ is the bank angle, and ψ is the heading angle of the ownship.

After the aircraft execute an advisory, a normally distributed noise with a standard deviation of 4° will be added to the bank angle, and a normally distributed noise with a standard deviation of $2m/s$ will be added to the speed. Similarly for the intruder aircraft, a normally distributed noise with a standard deviation of $10m$ will be added to its position at each time step. The noises here aim to account for the uncertainties in the environment and aircraft dynamics.

3.2.2.4 Terminal State

For the consideration of safety, the conflict is defined to be when the distance of two aircraft is less than a minimum separation distance $r^{min} = 0.3$ nautical miles [Bosson and Lauderdale (2018)]. This separation standard was chosen using the definition of well clear for Unmanned Aircraft Systems (UAS) according to Cook and Brooks [Cook and Brooks (2015)]. For large UAS in high-altitude airspace, the Horizontal Miss Distance (HMD) is defined to be 0.66nmi. For small UAS (55lbs vehicle or less) in low-altitude controlled airspace around airports, the horizontal separation is set to be a HMD of 0.36nmi. Using those values as reference, the nominal spatial separation standards picked for this UAM application are set to 0.3nmi horizontally. These are tighter than UAS standards because it is assumed that enhanced equipage capabilities will be installed onboard UAM aircraft [Bosson and Lauderdale (2018)].

Based on the above separation requirements, the terminal state of this MDP includes three different types of states:

- The distance from ownship to any intruder is less than r^{min} (referred to as a conflict state in the following);
- The ownship flies out of the map (referred to as a boundary state in the following);
- The ownship reaches the goal position (referred to as a goal state in the following).

To test whether a state is a terminal state, we can examine the state information. Suppose the current state (consider one intruder aircraft case) is

$$(i_x, i_y, i_{vx}, i_{vy}, o_x, o_y, o_{vx}, o_{vy}, o_\psi, g_x, g_y)$$

Then this state is a conflict state if

$$\sqrt{(i_x - o_x)^2 + (i_y - o_y)^2} < r^{min} \tag{3.4}$$

It is a boundary state if the current position of ownship is out of the map. The goal state can be determined similar to the conflict state by calculating the distance from the ownship to the goal position.

3.2.2.5 Reward Function

The goal in this dissertation is to make an aircraft quickly reach its destination and avoid potential conflict. These two objectives can be captured in the reward function defined as follows:

$$R(s) = \begin{cases} 1, & \text{if } s \text{ is goal state,} \\ 0, & \text{otherwise.} \end{cases} \tag{3.5}$$

With this reward setting, reaching a conflict state or a boundary state before the goal state will terminate the whole process with a reward of 0. Reaching a goal state will terminate this process with a reward of 1. So when maximizing the reward, the agent will try to reach the goal state and avoid conflict states and boundary states. Therefore we do not need to introduce a penalty for boundary states or conflict states.

3.3 Solution Method

In this section, we will introduce our proposed solution approach and the baseline method. We will describe how to apply these methods to solve our problem.

3.3.1 MCTS Algorithm

For the MDP formulated above, the most popular algorithm in the MCTS family, the Upper Confidence Bound for Trees (UCT) [Kocsis and Szepesvári (2006)], is used to solve this problem. UCT has some promising properties: it's very efficient and guaranteed to be within a constant factor of the best possible bound on the growth of regret (the regret is the expected loss due to not selecting the best action), and it can balance exploration and exploitation very well [Kocsis and Szepesvári (2006)].

In the MCTS algorithm (Figure 2.3), the nodes in the search tree denote the states in the state space of the MDP problem formulated in Section III. In the remaining part of this dissertation, the state and the node will be used interchangeably. The child nodes of a node are all the possible next states (nodes) resulting from different actions from the current state (node). Since there are 9

actions in the action space at each time step, each node will have at most 9 child nodes by executing these 9 different actions.

MCTS algorithm selects actions by lookahead search. Each edge (s, a) of the search tree stores an action value $Q(s, a)$ and its visit count $N(s, a)$. The tree is traversed by simulation, starting from the root state, which is the current state we are considering.

In **selection** step, the ownship will select a child node with maximum value in Equation 3.6, so as to maximize the mean action value $\overline{X}_j$ plus a bonus:

$$UCT = \overline{X}_j + 2C\sqrt{\frac{2\ln n}{n_j}} \tag{3.6}$$

Here the first term $\overline{X}_j$ is referred as exploitation term, which is directly from the formula

$$\overline{X}_j = Q(v)/N(v) \tag{3.7}$$

where the number $N(v)$ is the times this child node has been visited before and the value $Q(v)$ is the total reward of all playouts that passed through this child node (so that $Q(v)/N(v)$ is an approximation of the child node's state-action value). The second term $2C\sqrt{2\ln n/n_j}$ is referred as an exploration term where n is the number of times the current (parent) node has been visited, n_j is the number of times child j has been visited, and C is a constant to balance the exploration and exploitation. A higher C value will emphasize exploration and a lower C value will encourage exploitation. It should be noted that the value of C depends on the value scale of $\overline{X}_j$. Since the value of $C = 1/\sqrt{2}$ was shown by Kocsis and Szepesvari to satisfy the Hoeffding inequality with rewards in the range $[0, 1]$ [Kocsis et al. (2006)], it is reasonable to set $C = 1/\sqrt{2}$ in this dissertation. With a reward range different than $[0, 1]$, a different value of C may be needed.

If more than one child node has the same maximal value, the tie is broken randomly [Kocsis et al. (2006)]. It is generally understood that $n_j = 0$ yields a UCT value of ∞, so that if a node is never visited previously, it will be assigned to the largest possible value, to ensure that every child will be considered at least once before any expansion [Browne et al. (2012)]. This is the strategy used in this dissertation.

The second step for the UCT algorithm is **expansion**, which happens when the ownship is at a new node which it has never visited before. This step is adding this new node to the current tree under its parent node (the previous state), and setting its visiting number to 1 and cumulative reward to 0.

The third step is **roll out**, which aims to estimate the value for the newly added state in the expansion step. After a new node is added to the tree, its value will be determined by running a simulation to a terminal state following a random policy, until reaching a terminal state with a final reward. It should be noted that simulating to a terminal state usually requires many steps, which is time-consuming, and we will address this limitation of the MCTS algorithm in the next subsection "Estimated Value Function".

After an action is selected, the next state will be determined as follows:

For the intruder aircraft, its velocity remains unchanged since it's assumed that intruder aircraft can only fly straight at a fixed velocity. The position of the next state is the sum of the current position and current velocity, and a Gaussian noise with a standard deviation of 10m.

For the goal position, it remains unchanged if the ownship does not arrive at the goal state. If the ownship arrives at the goal state, this state will be a terminal state with a reward of 1.

For the ownship, the next state is harder to determine since it also depends on the selected action (the changing of bank angle and speed), and we will use the following difference equations (discretization of the differential Equation 3.3 using Forward Euler method) with uncertainty incorporated to update the next state according to current state and current action:

Assume there is only one intruder aircraft and the current state is

$$(i_x, i_y, i_{vx}, i_{vy}, o_x, o_y, o_{vx}, o_{vy}, o_v, o_\psi, o_\phi, g_x, g_y)$$

and the selected action is $a = (a_\psi, a_s)$, then the next state

$$(i'_x, i'_y, i'_{vx}, i'_{vy}, o'_x, o'_y, o'_{vx}, o'_{vy}, o'_v, o'_\psi, o'_\phi, g'_x, g'_y)$$

can be determined as follows:

$$
\begin{aligned}
i'_{vx} &= i_{vx} \\
i'_{vy} &= i_{vy} \\
i'_{x} &= i_{x} + i_{vx} + \epsilon \\
i'_{y} &= i_{y} + i_{vy} + \epsilon \\
o'_{\phi} &= o_{\phi} + a_{\phi} \times \Delta t + \epsilon \\
o'_{v} &= o_{v} + a_{a} \times \Delta t + \epsilon \\
o'_{\psi} &= \frac{g \cdot \tan \phi}{o_v} \\
o'_{vx} &= o_v \cos o_{\psi} \\
o'_{vy} &= o_v \sin o_{\psi} \\
o'_{x} &= o_x + o_{vx} \\
o'_{y} &= o_y + o_{vy} \\
g'_{x} &= g_x \\
g'_{y} &= g_y
\end{aligned}
\tag{3.8}
$$

where $\Delta t = 1s$ is the fixed time step and ϵ is the noise described in Section III.B.3. Note that the ownship will make a decision every 5 seconds (5 time steps) and maintain its action during these 5 time steps.

The final step of MCTS is **backpropagation**. After simulating the whole process to a terminal state, the final reward and visit count of all traversed edges are updated. Each traversed edge accumulates the reward and increases the visit count by 1, and we can get the mean state-action value from the total reward and visit count.

One iteration of the above four steps is called one simulation. If the computation budget allows (e.g., the decision needs to be made in 100ms), sufficient simulations will be repeated, which can provide a good approximation for the values of different nodes. When the simulation stops and a decision needs to be made, the most promising node will be selected by performing exploitation (set $C = 0$ in Equation 3.6).

Estimated Value Function

The only remaining concern now is that the ownship may not have sufficient time to run this algorithm, since the algorithm presented in this dissertation is an online algorithm, which means it is vital for the ownship to compute quickly to make decisions. Since simulating this process to a terminal state usually needs many steps, simulating this process to a fixed search depth d is beneficial to reduce computation time in the roll out step. More specifically, if the algorithm simulates to a fixed search depth d and reaches a non-terminal state, the agent will use the estimated value function as the terminal reward and backpropagate this reward information. An example of the building of state-action decision tree is given in Figure 3.2, where the search depth is fixed at 2. For the estimated value function, intuitively, if at a state where the ownship is closer the goal state, this state should be a better state without any other information, so the following estimated value function is used for the non-terminal states, so that the ownship can judge the goodness of any non-terminal state:

$$\tilde{V}(s) = 1 - \frac{d(o,g)}{\max d(o,g)}, \text{ if } s \text{ is non-terminal state} \tag{3.9}$$

where $d(o,g)$ denotes the distance from ownship to goal position. $\max d(o,g)$ is the maximum distance from ownship to the goal state, which is the diagonal distance of the map (if the map has an irregular convex shape, we can use the diameter of this convex shape). In this way, if there is no conflict with intruder aircraft or the border (which has reward 0), the ownship will get a positive reward between 0 and 1, depending on how far the ownship is from the goal state.

The above procedure is summarized in Algorithm 1. In this pseudo code, we use v to denote the node and s to denote the state. $s(v)$ means the state of a node and $v(s)$ means the node created from state s. $Q(v)$ is the total reward of all playouts that passed through the node v and $N(v)$ is the times the node v has been visited before. $d(v)$ represents the search depth of the node v.

3.3.2 Optimal Reciprocal Collision Avoidance (ORCA) Method

A popular approach to this computational guidance problem with the separation assurance capability problem is the Optimal Reciprocal Collision Avoidance (ORCA) Method [Van Den Berg

Algorithm 1 MCTS-UCT algorithm

function UCTSEARCH(s_0)
    create root node v_0 with state s_0
    while within computational budget **do**
        $v_l \leftarrow$ TREEPOLICY(v_0)
        $\Delta \leftarrow$ DEFAULTPOLICY($s(v_l)$)
        BACKUP(v_l, Δ)
    return a(BESTCHILD($v_0, 0$))

function TREEPOLICY(v)
    while v is nonterminal **and** $d(v) \leq d$ **do**
        if v not fully expanded **then**
            return EXPAND(v)
        else
            $v \leftarrow$ BESTCHILD(v, C)
    return v

function EXPAND(v)
    choose $a \in$ untried actions from $A(s(v))$
    $s(v') =$ PROCEED($s(v), a$)
    add the new child v' to v
    return v'

function BESTCHILD(v, c)
    return $\underset{v' \in \text{children of } v}{\operatorname{argmax}} \frac{Q(v')}{N(v')} + c\sqrt{\frac{2 \ln N(v)}{N(v')}}$

function DEFAULTPOLICY(s)
    while s is nonterminal **and** $d(v(s)) \leq d$ **do**
        choose $a \in A(s)$ uniformly at random
        $s \leftarrow$ PROCEED(s, a)
    return reward for state s

function BACKUP(v, Δ)
    while v is not null **do**
        $N(v) \leftarrow N(v) + 1$
        $Q(v) \leftarrow Q(v) + \Delta$
        $v \leftarrow$ parent of v

function PROCEED(s, a)
    $s' \leftarrow$ next state from current s, a
    $d(v(s')) \leftarrow d(v(s)) + 1$
    return s'

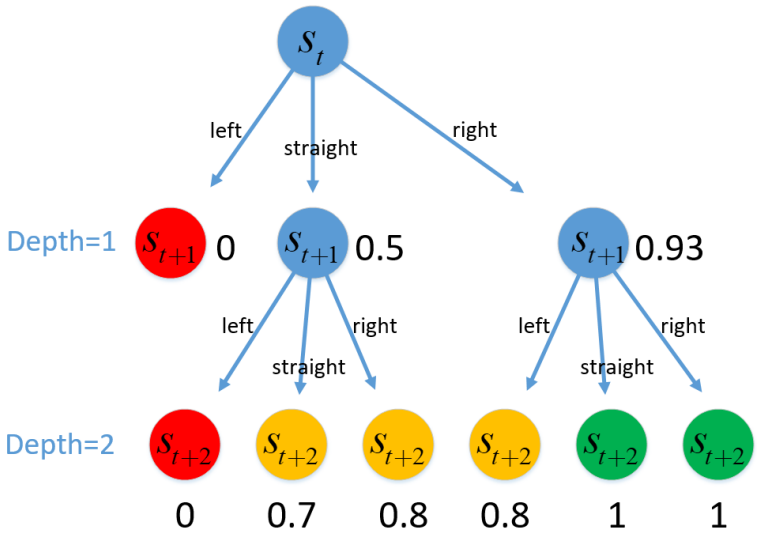


Figure 3.2: Illustration of the state-action tree built in the MCTS algorithm with search depth 2. For illustration purposes, we only consider 3 actions in this case: turn left, turn right, and go straight. Here red node denotes conflict state, green node denotes goal state. Yellow nodes mean that the agent simulates to depth 2 and uses the estimated value function as the final reward for the non-terminal state, depending on the distance between ownship and goal position. The state-action value of each node at time step $t+1$ is the average of all its child node values. Based on this illustration, the agent will select to turn right at current state s_t.

et al. (2011)]. The ORCA method is a very efficient algorithm and the safety and reliability of the ORCA method have already been demonstrated in realistic applications [Alejo et al. (2014); Conroy et al. (2014)]. Based on the Velocity Obstacles (VO) [Fiorini and Shiller (1998)] concept, it provides a sufficient condition for multiple robots to avoid collisions among one another, and thus can guarantee collision-free navigation. Similar to our algorithm presented above, the ORCA method is also a reaction-based method which specifies one-step interaction rules for the current geometric configuration. In order to further understand the performance of the MCTS approach, we present here how to use the ORCA method to solve this problem. After comparing the performances of MCTS and ORCA, the benefits and limitations of the proposed MCTS approach will be discussed.

The basic idea of ORCA is that, at each time step, the ownship first decides the set of the velocities which is conflict-free with all the other aircraft for at least a preset amount of time τ, when the intruder aircraft would continue to move at their current velocity, and we denote this set

$ORCA^{\tau}$. Next, the ownship will select its new velocity as close as possible to its preferred velocity (which is the velocity point directly to its destination) in the set $ORCA^{\tau}$.

Since in our problem, we are only controlling one ownship aircraft. So in contrast to the original ORCA paper [Van Den Berg et al. (2011)] where each agent will take half of the responsibility to remain on a collision-free trajectory, in this dissertation we let the ownship take the full responsibility when selecting the conflict-free velocity.

Besides, since we are controlling the eVTOL aircraft, the selected velocities also need to satisfy the dynamic constraint. Specifically, we restrict the ownship to select the velocity with speed in the range $(v - 5m/s, v + 5m/s)$ where v is the current speed of the ownship. Here the number $5m/s$ was chosen to be consistent with the MCTS algorithm, where we allow the ownship to change speed $5m/s$ per second. One drawback of the ORCA algorithm is that it always selects the velocity directly and cannot incorporate the bank angle model in a straightforward way. So here we restrict the ownship to select velocity in the 10-degree field of view, which means the ownship can change its heading angle less than $10°/s$. This dynamics constraint is shown in Figure 3.3, where the ownship will select a velocity in the shaded area, which we denote as set S. In this figure, θ is 20°, v^{min} and v^{max} are chosen to be $v - 5m/s, v + 5m/s$ where v is the current speed of the ownship. Now the goal of ORCA algorithm is to solve this optimization problem: choose a velocity which minimizes the distance to the preferred velocity, and subject to the ORCA constraint (the new velocity should be in the intersection of all the safe sets) and dynamics constraint:

$$v^{new} = \underset{v \in ORCA^{\tau} \bigcap S}{\operatorname{argmax}} ||v - v^{pref}|| \tag{3.10}$$

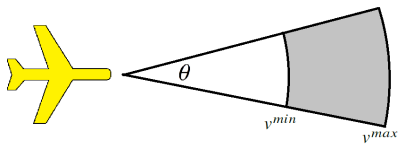


Figure 3.3: Sample area of the aircraft for ORCA algorithm, which we denote as set S.

Since we are considering the dynamics constraint which makes the feasible action space non-convex (shown in the shaded area of Figure 3.3), the solution approach using linear programming in the original paper [Van Den Berg et al. (2011)] is not applicable. We solve the above optimization problem by the following approach. After calculating the set $ORCA^{\tau}$, we sample N points uniformly in the set S (the shaded area in Figure 3.3). Then among all the sampled points, we will choose one point in set $ORCA^{\tau}$ that is closest to the preferred velocity, as shown in Equation 3.10]. However, when the density of air traffic is very high, the intersection of $ORCA^{\tau}$ and S may be empty. In this case, we will select the "safest possible" velocity for the ownship among the N sampled points, i.e. the velocity that minimally "penetrates" the constraints in set $ORCA^{\tau}$. For the number of sampled points, when the number N is larger, the final selected velocity will be closer to the optimal solution in this optimization problem.

Here we note the differences between the MCTS algorithm and the ORCA algorithm. First, the ORCA algorithm can select any velocity in the shaded area while the MCTS algorithm can only choose from 9 discrete actions at each time step. We will show the reduced action space compared to ORCA does not hurt the performance of the MCTS algorithm. Second, since the ORCA algorithm does not have a penalty for the ownship flying out of the map, we allow the ownship to fly out of the map and then fly back while implementing the ORCA algorithm. Third, since the original ORCA algorithm is proposed in deterministic case and we are introducing uncertainties in this dissertation, we increase the conflict radius of each aircraft to give the ORCA an extra buffer to avoid conflicts between aircraft.

Besides, when implementing the MCTS algorithm and ORCA algorithm, we only consider the nearby intruder aircraft (the ownship can only know the position and velocity of the intruder aircraft within 4000 meters), since the intruder aircraft far away will not affect the current decision of ownship while considering potential conflicts. We observe that this can speed up the algorithm tremendously, and we will present the result performance of these two algorithms in the next section.

3.4 Numerical Experiments

3.4.1 Simulator

To test the performance of the proposed algorithm and the baseline algorithm, a simulator was built in Python where the aircraft can fly freely in the two-dimensional en route airspace. The airspace has 24km length and 24km width, which is designed for future Urban Air Mobility free flight operations. The assessment of the algorithm involves running 1000 episodes in this simulator and then take the average of the statistics as the algorithm performance. Here one episode means the ownship flying from its initial position to a terminal state.

At the beginning of each episode, the initial position of ownship is at the bottom right corner of the map, and the initial speed of the ownship is set to $60m/s$, the initial flight direction is set to point directly to the center of the map. Then a fixed number of intruder aircraft are generated with speed uniformly distributed between $50m/s$ and $80m/s$ and heading angle uniformly distributed between 0° and 360°. The goal position of the ownship is also uniformly generated in the map.

Then the simulator will work as follows. Based on the assumptions mentioned before, the intruder aircraft can only fly at fixed velocity with a fixed direction. When any intruder flies out of the map, a new intruder will be randomly generated using the parameters setting described above so that the number of intruder aircraft is fixed. When there is a conflict between ownship and any intruder, the counter for the number of conflicts between ownship and intruder will increase by one. The conflicts between intruders are not addressed in this chapter. In addition, when we generate new intruder aircraft, the intruder should not appear too close to ownship, in which case the ownship might not be able to avoid this intruder no matter what action it takes. When the ownship reaches the goal state or flies out of the map or has a near mid-air collision with any intruder aircraft, this episode will end and a new episode will start. The near mid-air collision (NMAC) standard is defined to be 500 feet by the Aeronautical Information Manual (7-6-3) [Federal Aviation Administration (2017)]. Note in this simulator, when a conflict happens, we do not terminate this

episode but just record this conflict. This is because in a conflict state, the MCTS algorithm can still work to guide ownship escaping this conflict and avoid potential NMAC.

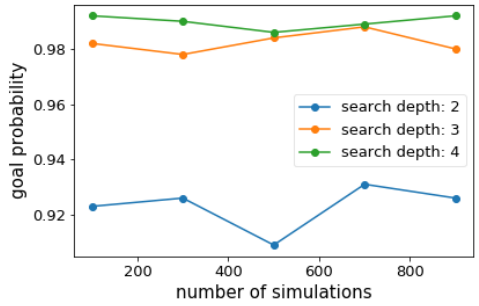
We also note that there is a simulator from NASA named Fe3 (Flexible engine for Fast-time evaluation of Flight environments). It was used to test the performance of different low-altitude high-density air traffic operations [Xue and Rios (2017); Bulusu et al. (2018); Xue et al. (2018)]. In their simulator, the goal position and origin are on the boundary of the map. This simulator is more suitable for the case where big airspace was divided into several small sectors and we only need to control aircraft in one small sector to guarantee there is no conflict between aircraft. When the aircraft exit the current sector and move into the next sector, another controller will take over this aircraft. For the simulator used in this dissertation, it can be used in a scenario where bounded airspace contains both static obstacles (walls or geofences) and dynamic obstacles (intruders or birds), such as airspace above a small city. In this case, the goal position and origin will be in the same airspace region.

3.4.2 Results

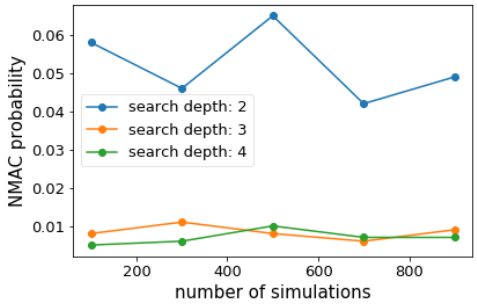
In the following experiments, we run 1000 episodes in the simulator for each algorithm with different parameter settings, then we record and compare the percentage of these 1000 episodes where the ownship reaches the goal state successfully, the percentage of episode termination due to a NMAC, average conflicts in each episode, and average running time for each decision making step.

3.4.2.1 Performance of MCTS algorithm with different parameters

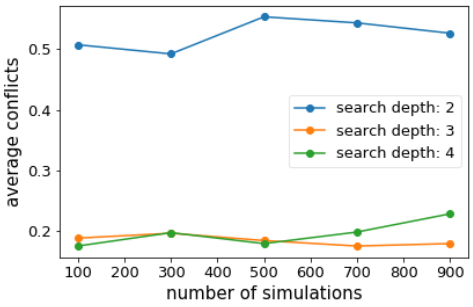
In the MCTS algorithm, there are two parameters that are important to the performance of this algorithm: the number of simulations n that each time a decision needs to be made and the search depth d, which is discussed in Section IV. In this experiment, the number of intruders is varied from 10 to 80 with step 10, the number of simulations is varied from 100 to 900 with step 200, and



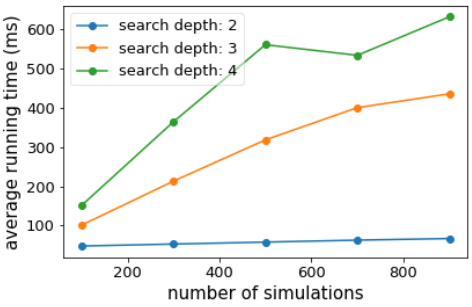
(a) Probability of reaching goal state



(b) Probability of having a NMAC



(c) Average conflicts in each episode



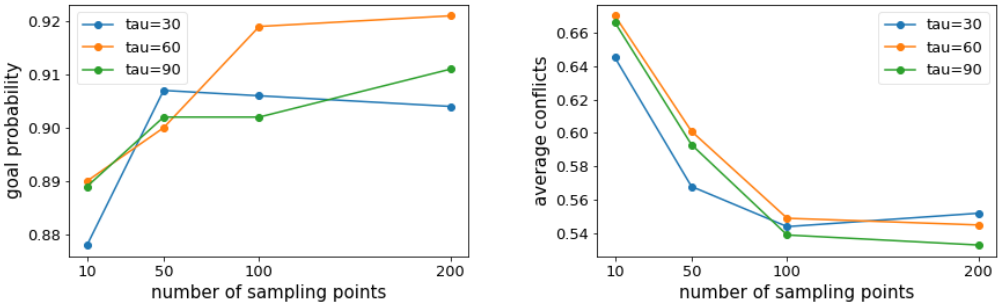
(d) Average running time for each decision

Figure 3.5: Performance of MCTS algorithm with different parameters when there is 80 intruder aircraft.

the search depth is chosen from 2, 3, and 4. Based on the results and performance comparison, the best parameters n and d are chosen to conduct the second experiment.

Since the performance trend of the MCTS algorithm is consistent for different numbers of intruder aircraft, in Figure 3.5 we only show the result where the number of intruder aircraft is 80, and the results with other numbers of intruder aircraft can be found in the Appendix.

Figure 3.5 (a), (b) and (c) show that the search depth $d = 2$ performs worse than deeper search depths (the ownship reaches fewer goal states and has more NMACs and conflicts). This is because when the search depth is deeper, the ownship can look further in the future and thus can take action to avoid the conflicts foreseen in the further future. The search depth $d = 3$ and $d = 4$ do not show much difference from the results. They perform equally well. From Figure 3.5 (a), (b) and (c), we also observe that the number of simulations does not affect the performance much,



(a) Probability of reaching goal state (b) Average conflicts in each episode

Figure 3.7: Performance of ORCA algorithm with different parameters when there is 80 intruder aircraft.

which implies that in this problem, MCTS algorithm does not need too many simulations to have an accurate approximation of the action value.

The average running time in Figure 3.5 (d) is growing linearly with the number of simulations and the search depth is also the main factor slowing down the algorithm because the search tree is deeper.

Based on the results in Figure 3.5, since increasing the search depth d to 4 and increasing the number of simulations will increase the computation time while cannot bring much performance improvement, the number of simulations $n = 100$ and search depth $d = 3$ were chosen for the next experiment to compare with ORCA algorithm.

3.4.2.2 Performance of ORCA algorithm with different parameters

For the ORCA algorithm, the preset time τ decides how long the agent can "see" in the future, and the number of sampling points N decides the optimality of the selected velocity. In this experiment, we did a stress test (maintain the number of intruder aircraft as 80) for these two parameters with different values, by setting τ to 30, 60, 90 and N to 10, 50, 100, 200. The result is shown in Figure 3.7.

From Figure 3.7 we can see that with a larger number of sampling points, the performance is better with more reached goals and fewer conflicts. However, the ORCA algorithm does not show

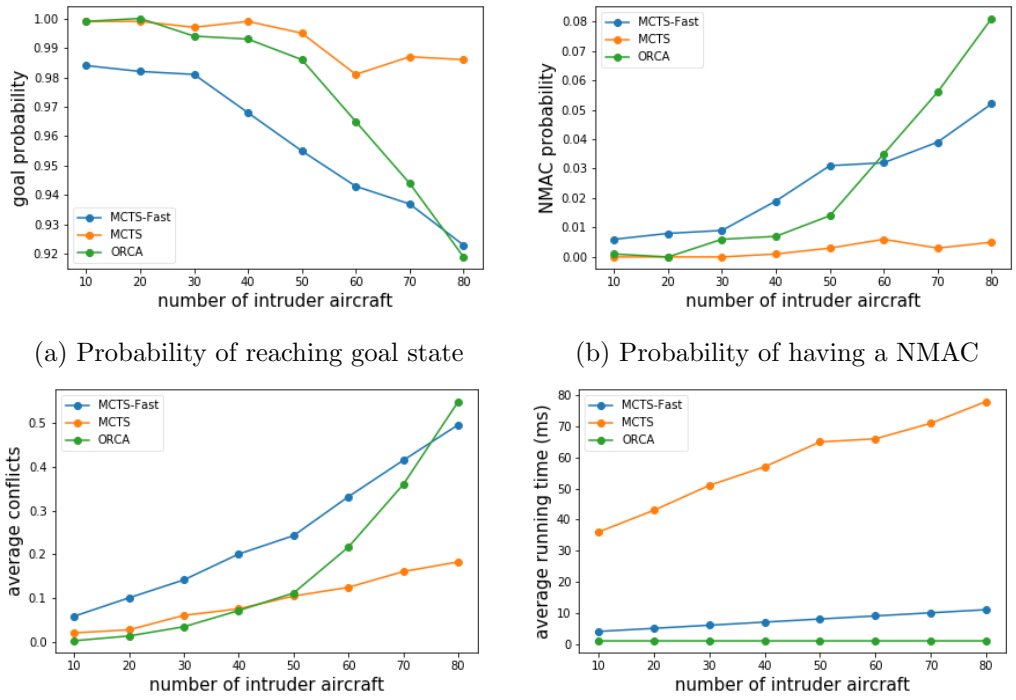
much improvement by increasing N from 100 to 200. Since the computation time is increasing linearly with N, we choose N to be 100 for the next experiment. Also, we choose the parameter τ to be 60 since from the result it performs slightly better than the other two in terms of the reached goals.

3.4.2.3 Comparison between MCTS, MCTS-Fast, ORCA algorithm

In this experiment, we change the number of intruder aircraft from 10 to 80 with step 10 and compare the performance of the MCTS algorithm and ORCA algorithm, with the parameter values selected from the above experiments. Besides, we also examine the performance of the MCTS algorithm in an extreme case where the decision time is very short. In this extreme case, we only allow the MCTS to run 10 simulations to build the search tree and then select an action. We call this variant of the MCTS algorithm "MCTS-Fast". We generate 1000 same episodes for the three algorithms and Figure 3.9 shows the performance results of algorithm MCTS, MCTS-Fast, and ORCA.

The three algorithms in Figure 3.9 show similar patterns: with the increasing number of intruders, the number of conflicts and probability of having a NMAC are both increasing, and the probability of reaching goal state is decreasing. For all three algorithms, the ownship can reach the goal state with more than 90% in the 1000 episodes and the average conflicts in each episode are under 0.6, which means all three algorithms are very effective at avoiding potential collisions. In addition, when the intruder number is below a certain threshold (approximately under 40), the ORCA and MCTS algorithms can help the ownship reach goal state without encountering any NMAC for more than 99% of the time. Also, with a restricted computation time, the MCTS-Fast algorithm shows a little performance loss, which demonstrates the robustness of the MCTS algorithm.

As shown in Figure 3.9 (b) and (c), in terms of the percentage of NMAC episodes and the number of average conflicts, the MCTS algorithm performs better than the ORCA algorithm, especially when the intruder aircraft number is large (air traffic density is high). The reason is that at each time step, ORCA will select a velocity only based on the current state without considering



(a) Probability of reaching goal state

(b) Probability of having a NMAC

(c) Average conflicts in each episode

(d) Average running time for each decision

Figure 3.9: Performance of MCTS, ORCA, MCTS-Fast with different number of intruder aircraft.

the action for the next state, while the MCTS algorithm can look ahead for 3 steps (search depth is 3). When there are 9 actions to choose at each time step, the ownship can estimate the outcome up to $9^3 = 729$ different action combinations (similar to the illustration in Figure 3.2, where the ownship can explore 7 different action combinations with search depth 2).

Figure 3.9 (d) shows that the computation time of all three algorithms is growing linearly with the increasing number of intruder aircraft. On average, the MCTS algorithm needs around 50ms to 60ms to generate the output action (the running time under 100ms is acceptable for a real-time decision-making system, given that the decision is made every 5 seconds), while ORCA algorithm only needs 2ms to finish the computation. This shows that the improved performance of the MCTS algorithm comes at the price of longer computation time.

3.4.3 Limitations

One limitation of the MCTS algorithm is that it cannot be applied in multi cooperative aircraft case in a straightforward way. If we want to apply this MCTS algorithm to all the aircraft flying in the airspace, some communication between them is needed since the value of action for one aircraft also depends on the action of other aircraft.

Besides, in this work we assume full observability, where the ownship can sense the intruder aircraft information (position and velocity) perfectly. However in practice when the state information becomes imperfect, some techniques such as Kalman Filter or Partially Observable MDP will be necessary to filter out the sensor noise.

We also observe another limitation of the MCTS algorithm through the numerical experiment. In Figure 3.10, we plot the resulted trajectories by following the actions from the MCTS algorithm and ORCA algorithm when there is no intruder aircraft blocking the way to the goal position. In this case, the ORCA algorithm can select a velocity pointing directly to the goal position since the action space for the ORCA algorithm is continuous (the shaded area in Figure 3.3). MCTS algorithm only has 3 fixed turning rate of bank angle, which makes it difficult to point directly to

the goal position. This will make the ownship keep changing the heading angle when approaching the goal with extra fuel cost.

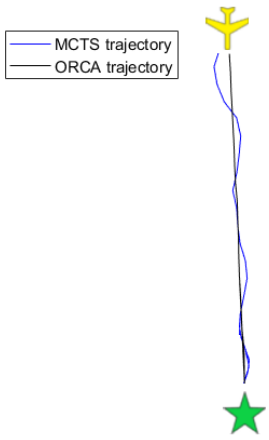


Figure 3.10: The trajectories generated by MCTS algorithm and ORCA algorithm when there is no intruder.

3.5 Conclusion

A shift from human-centric air traffic control systems towards higher levels of autonomy is required to enable safe and efficient Urban Air Mobility operations. In this chapter, we proposed a computational guidance algorithm with separation assurance capability for autonomous free flight operations, which can guide a single aircraft to its destination through controlling the bank angle and acceleration, while maintaining safe separation with other intruder aircraft. The problem is formulated as a Markov Decision Process (MDP) with uncertainty in aircraft dynamics and the environment. The formulated MDP is solved by Monte Carlo Tree Search (MCTS) algorithm with an estimated value function to reduce the computation time. Numerical experiments in the airspace simulator show that our algorithm has promising performance and outperforms the baseline algorithm Optimal Reciprocal Collision Avoidance (ORCA) in dense air traffic scenarios.

CHAPTER 4. SCALABLE MULTI-AGENT COMPUTATIONAL GUIDANCE WITH SEPARATION ASSURANCE FOR AUTONOMOUS URBAN AIR MOBILITY OPERATIONS

4.1 Introduction

In this chapter, we address the research problem 2 listed in Section 1.2. Comparing with the previous chapter, here we propose a message-passing based decentralized computational guidance algorithm with a separation assurance capability that can scale to multiple cooperative aircraft, where we formulate this problem as a Multiagent Markov Decision Process (MMDP) and solve this formulated MMDP using Monte Carlo Tree Search (MCTS) algorithm. We use a logit level-k model for the multi-aircraft coordination based on the message-passing mechanism through wireless communication. To achieve higher scalability, we introduce the airspace sector concept into the UAM environment by dividing the airspace into sectors, so that each aircraft only needs to coordinate with aircraft in the same sector. A high-density free-flight airspace simulator in the OpenAI Gym environment is built to validate and demonstrate the performance of the proposed algorithm. Through numerical experiments over several case studies in the free flight simulator with environment uncertainty, the proposed algorithm shows promising performance. Additionally, we also performed a stress test on the roundabout test problem, which consists of making a certain number of aircraft fly to the diametrically opposed point at a common speed on an annulus.

In this chapter, we first describe the problem and formulate it as Multiagent MDP. In section 4.3 we present the designed MCTS algorithm to solve the formulated multiagent problem. The numerical experiment and results are shown in section 4.4. Section 4.5 concludes this chapter.

4.2 Problem Formulation

4.2.1 Problem Statement

The goal of this research is to develop a decentralized algorithm that runs on each aircraft to provide tactical guidance commands. These commands will help each aircraft arrive at their respective destinations while avoiding potential LOS events between them during the flight. In this dissertation, we assume a free flight concept of operations (ConOps), where each flight's trajectory is free of airspace constraints. Here we investigate the feasibility of the free flight ConOps and push its safety limits with new tools from MMDP and MCTS. However, we would like to identify other approaches and ConOps such as pre-departure flight planning for strategic deconfliction, and/or implementing structured airspace to ensure a higher level of safety. In the MMDP formulation, at each decision point, the vehicle's action is decided directly from the state, which incorporates all the information (the position and velocity of all the intruder aircraft and the destination/goal of ownship) to decide which action is optimal for the corresponding state.

When controlling the aircraft, only horizontal actions are considered, which means all the aircraft will be flying at the same altitude and this problem can be solved in two dimensions. This assumption makes it possible to incorporate multiple flight levels to deal with the high-density air traffic in UAM.

The objectives for this specific MMDP problem are two-fold: the first is to avoid potential LOS events among all the moving aircraft, the second is to guide all the aircraft to their destinations as quickly as possible during the flight. Therefore, the reward function needs to capture both two objectives.

Based on the above description, this problem will be mathematically formulated as a MMDP problem in the next subsection.

4.2.2 MMDP Formulation

The MMDP formulation in this chapter is similar to the last chapter and here we will briefly introduce it.

4.2.2.1 State Space

The state includes the necessary information for the algorithm to issue actions to all the aircraft: the position, speed, heading angle, and goal position for all of the aircraft. More specifically, the state for one aircraft is $(x, y, v, \psi, g_x, g_y)$, where $(x, y), v, \psi$ are the position, speed, and heading angle for the aircraft, and (g_x, g_y) is the goal position for this aircraft. Stacking the information for all of the controlled aircraft, the state space for the MMDP becomes a $n \times 6$ matrix, where n is the number of aircraft and each row of the matrix represents the information for one aircraft.

4.2.2.2 Action Space

At each time step (5 seconds), the aircraft can choose to turn its heading at a certain rate. More precisely, the advisory of heading angle for each aircraft constitutes the action set $\mathcal{A} = \{-5^\circ/s, 0^\circ/s, +5^\circ/s\}$ where positive corresponds to the right turn and negative to left turn. The changing rate of heading angle is determined assuming the aircraft is flying with cruise speed at $190km/h$ [Airbus (2018)] and banking angle at 25° (the banking angle limit is chosen to be 25° for the passenger comfort consideration). At each time step, the proposed algorithm will run onboard to issue one action from the action set for the aircraft based on the current state. After the algorithm running, the aircraft will maintain the issued action during this time step.

It's natural to consider extending the set of actions (conflict resolution advisories) to include more options such as vertical resolution and speed resolution. However, using the MCTS algorithm to calculate the optimal action will be more time consuming with the extended action space since the tree size will grow exponentially with the number of actions. Because computation time is an important factor for the real-time onboard algorithm, some techniques are necessary for extending action space in future steps, such as truncated Monte Carlo search algorithms [Tesauro and Galperin (1997)], progressive strategies for MCTS [Chaslot et al. (2008); Coulom (2007); Wang et al. (2009); Couëtoux et al. (2011)], or using a policy network to narrow down the search to high-value actions [Silver et al. (2016)]. In this dissertation, we use this defined action space to keep our scope more focused.

4.2.2.3 Dynamical Model

Based on the current state and current action, Dubin's kinematic model will be used to compute state transition for each aircraft:

$$\dot{x} = v\cos\psi \tag{4.1}$$

$$\dot{y} = v\sin\psi \tag{4.2}$$

$$\dot{\psi} = a_{\psi} \tag{4.3}$$

where v is the cruise speed, ψ is the heading angle, and a_{ψ} is the selected action describing the changing rate of heading angle for one aircraft. Following the aircraft performance data from Airbus Vahana [Airbus (2018)], the cruise speed of the aircraft is set to $190km/h$, and we restrict the speed to be in between $v_{\min} = 165km/h$ and $v_{\max} = 220km/h$ [Pradeep and Wei (2018a); Airbus (2018)] since there is uncertainty in the speed.

After an aircraft executes an advisory, the aircraft speed is held constant between decision stages with uncertainty, which is modeled as a Gaussian distribution centered on the aircraft's cruise speed with a standard deviation of $5m/s$. The changing rate of heading angle distribution is centered on the resolution advisory with a standard deviation of $2^{\circ}/s$. The noises here aim to account for the uncertainties in the environment and aircraft dynamics. In previous works [Ong and Kochenderfer (2016); Julian and Kochenderfer (2017)], the uncertainties are modeled as Gaussian distributions with a standard deviation of $2m/s$ for speed and 3° for heading angle when the banking angle is less than 25°. In this dissertation we increase the standard deviation for speed to $5m/s$ and lower the standard deviation for change of heading angle to $2^{\circ}/s$ to better model the uncertainties since Vahana aircraft is flying at a higher speed.

4.2.2.4 Reward Function

The focus of our computational guidance system is to achieve the dual objectives of maintaining safety while guiding the aircraft to their destinations as quickly as possible. These objectives are captured in a reward function composed of a sum of the reward function for each individual aircraft.

For the consideration of safety, the LOS event is defined to be when the distance of two aircraft is less than a minimum separation distance $r^{min} = 0.5$ nautical miles [Bosson and Lauderdale (2018)]. This separation standard was chosen using the definition of well clear for UAS according to Cook and Brooks [Cook and Brooks (2015)]. For large UAS in high-altitude airspace, the Horizontal Miss Distance (HMD) is defined to be 0.66nmi. For small UAS (55lbs vehicle or less) in low-altitude controlled airspace around airports, the horizontal separation is set to be a HMD of 0.36nmi. Using those values as reference, the nominal spatial separation standards picked for this UAM application are set to 0.5nmi horizontally. This value is tighter than UAS standards because it is assumed that enhanced equipage capabilities will be installed on the eVTOL aircraft [Bosson and Lauderdale (2018)].

Based on the above separation requirements, we define the following two different types of states:

- The distance between two aircraft is less than r^{min} (referred to as a LOS state in the following);
- The aircraft reaches the goal position (referred to as a goal state in the following).

With the two types of states defined above, the reward function for one aircraft is defined as follows:

$$r(s) = \begin{cases} 1, & \text{if } s \text{ is goal state,} \\ 0, & \text{if } s \text{ is LOS state,} \\ 1 - \frac{d(o,g)}{\max d(o,g)}, & \text{otherwise.} \end{cases} \tag{4.4}$$

where $d(o, g)$ denotes the distance from the aircraft to its goal position. $\max d(o, g)$ is the maximum distance from an aircraft to its goal position, which is the diagonal distance of the map (if the map has an irregular convex shape, we can use the diameter of this convex shape). In this way, if an aircraft is not at LOS state (which has reward 0), this aircraft will get a positive reward between 0 and 1, depending on how far this aircraft is from the goal state.

The reward function for one aircraft is normalized to range $[0, 1]$ for the following reason. In MCTS, the algorithm will keep selecting the action with max value:

$$\bar{r}_j + 2C\sqrt{\frac{2\ln n}{n_j}} \tag{4.5}$$

where $\bar{r}_j$ is the mean reward value for action j, n is the number of times the current state has been visited, and n_j is the number of times action j has been selected during the process for building the search tree. In Equation 4.5, the first term describes the average value for an action from previous information, the second term measures the uncertainty of this action, and the coefficient C can balance these two terms. It should be noted that the value of C depends on the reward scale $\bar{r}_j$. The reward function is normalized to range $[0, 1]$ since for this reward range, the value of $C = 1/\sqrt{2}$ was shown by Kocsis and Szepesvari to satisfy the Hoeffding inequality [Kocsis et al. (2006)]. With a reward range different than $[0, 1]$, a different value of C will be needed.

After we have the reward function definition for one aircraft, the reward function for the MMDP is defined to be the sum of reward for each aircraft:

$$R(s) = \sum_i r_i(s) \tag{4.6}$$

where $r_i(s)$ is the reward function for aircraft i defined in Equation 4.4.

With this reward setting, when we solve the formulated MMDP by maximizing the reward, all of the aircraft will try to select action leading to a state that is closer to the goal position (which has positive reward) and avoid any LOS states (which has 0 reward). Since non-LOS states are always preferred than LOS states in the above reward setting, we do not need to introduce a penalty (such as a negative reward) for LOS states.

4.3 Solution Method

In this dissertation, we will use the most popular algorithm in the MCTS family, the Upper Confidence Bound for Trees (UCT) to solve the MMDP problem formulated above. The details of UCT algorithm implementation for the single aircraft case can be found in [Yang and Wei (2018)]. As formulated in Section III, computing the global optimal solution is impractical for more than

a few aircraft since the state space will grow linearly and the action space will grow exponentially with the increasing number of aircraft. Therefore we use a variant of logit level-k model [Stahl II and Wilson (1994); Stahl and Wilson (1995)] to solve the above issue. In the logit level-k model, an agent at logit level-k assumes all of the other agents follow logit level-$(k-1)$ strategy. In this way, the actions of other agents become fixed thus the tree search process can avoid the action explosion issue in the multi-agent case.

Cooperative Sequential Decision Making

In the logit level-k model, a level-0 agent selects actions randomly by following a uniform distribution. A level-1 agent assumes that all the other agents adopt level-0 strategies and selects actions according to the logit distribution

$$P(a_i) \propto e^{Q_1^*(s,a_i)} \tag{4.7}$$

where $Q_1^*(s, a_i)$ is the Q-function for the state-action pair (s, a_i) of a level-1 agent assuming other agents follow level-0 strategies. A level-k agent assumes that the other agents adopt level-$(k-1)$ strategies and select their own actions according to Equation 4.7.

In this dissertation we use a deterministic variant of the logit level-k model. More specifically, a level-0 aircraft selects the action to fly straight, and a level-1 aircraft assumes other aircraft adopt the level-0 strategy and selects action a^* with max Q value:

$$a^* = \underset{a}{\text{argmax}} Q_1^*(s, a_i) \tag{4.8}$$

where the Q-function $Q_1^*(s, a_i)$ is approximated from the MCTS algorithm.

The deterministic logit level-k model presented above updates the action for the next level in a synchronous way (e.g., the aircraft at the same level do not know the actions among each other). As shown in Figure 2.4, knowing the action of other aircraft is helpful for the action selection, therefore we update the action for the next level in an asynchronous way.

More specifically, suppose currently we have n aircraft flying in the air. Then at the beginning of the search algorithm when all the aircraft are at level 0, the joint actions $\mathbf{a}$ for all aircraft are

initialized to 0 at first:

$$\mathbf{a} = \{a_1, a_2, \cdots, a_n\} \tag{4.9}$$

where a_i is the action for aircraft i and $a_1 = a_2 = \cdots = a_n = 0°/s$.

Then starting from the first aircraft, each aircraft will run the algorithm onboard by building a search tree, assuming all of the other aircraft will take action according to the joint action $\mathbf{a}$ and follow the dynamical model described in Section III. B. 3. The searching process is similar to the process described in [Yang and Wei (2018)] and the difference is all aircraft can turn according to the joint action.

Next, assume the tree search result for the first aircraft is a_1^*, then the first aircraft will send this action information a_1^* to all the remaining aircraft through wireless communication and the joint action will be updated as follows:

$$\mathbf{a} = \{a_1^*, a_2, \cdots, a_n\} \tag{4.10}$$

Note here only the first aircraft is at level 1 and all the other aircraft are still at level 0. Then after receiving the action information from the first aircraft, the second aircraft begins running the MCTS algorithm onboard assuming all of the aircraft are following the most recent joint action. This process will iterate over all the aircraft until all the $a_1, a_2, \cdots, a_n$ are updated, when all the aircraft are at level 1. We can keep updating the actions for all of the aircraft to higher levels. Since numerical experiment results do not show much performance improvement for level 2 comparing with level 1, for the consideration of computation time, in this dissertation we stop this process after reaching level 1. After we have the level 1 joint action, all of the aircraft will execute this joint action for 5 time steps, after which all of the aircraft will run the algorithm again to generate new joint action.

4.4 Numerical Experiments

4.4.1 Simulator

To test the performance of the proposed algorithm, a simulator was built in Python where multiple aircraft can fly freely in the two dimensional en route airspace above New York City. We envision there will be multiple altitude levels where the eVTOL aircraft are operated. In the scope of this dissertation, we only focus on one altitude level (a two dimensional environment).

To validate the performance of this algorithm in real-world applications, we will simplify the UAM network by following the generic city model presented in [Kohlman and Patterson (2018); Patterson et al. (2018)]. In this generic city model, seven vertiports are distributed in a "six around one" hexagonal pattern. As shown in Figure 4.1, vertiport 1 is at in the center of the hexagon and located equidistant from the other six vertiports at a distance of 16km, which will cover the main congestion area of New York City. Overlays of the vertiport network are shown on a Google map image of New York in Figure 4.1, which shows typical New York traffic on a Friday at 5 pm [Google (2020)].

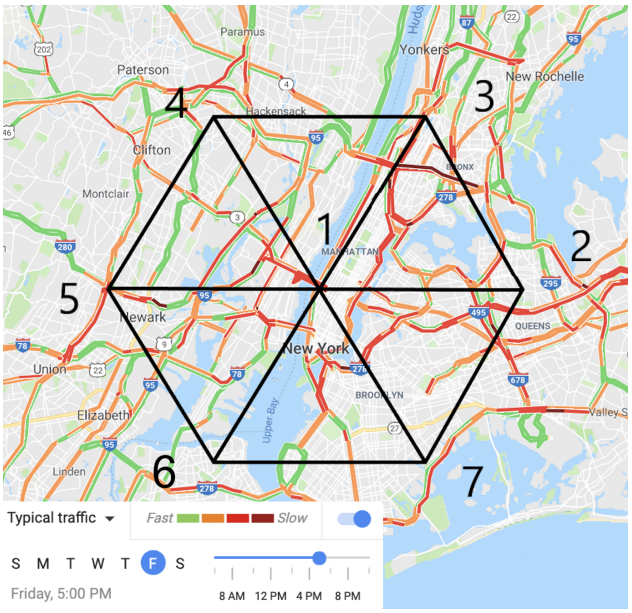


Figure 4.1: Network of seven vertiports overlaid on New York city with segment length 16km.

In real-world scenarios, the arrival and departure routes of large, conventional aircraft in terminal airspace near airports [Syed et al. (2017)], the presence of flight restrictions and restricted airspace [Vascik et al. (2019)], and the noise of the eVTOL vehicles [Thipphavong et al. (2018)] may all limit the UAM airspace shape and restrict the eVTOLs operations between vertiports. The proposed algorithm in this dissertation does not work under such scenarios yet, especially when the restricted airspace is in an irregular shape. A direction of future work would be to adapt the proposed algorithm to incorporate various airspace restrictions to make it more practical.

4.4.2 Airspace Sectorization

In the numerical experiment, the result shows the computation time for the algorithm grows linearly with the increasing number of aircraft, which makes it intractable for real-time guidance when the number of aircraft is over 15. To mitigate this issue, in this dissertation we introduce the concept of airspace sectorization to reduce computation time for the proposed algorithm by distributing the coordination workload to different sectors. As shown in Figure 4.2, the airspace with 7 vertiports is divided into 7 hexagonal sectors and the center of each hexagonal sector is one vertiport. One advantage of this hexagonal sector configuration is that hexagons can form a tessellation of a two dimensional airspace, which means this sector configuration can be easily extended to larger airspace.

In this sector setting, each aircraft only needs to coordinate with other aircraft in the same sector. More specifically, the aircraft in one sector only gather information of other aircraft in the same sector and the decision-making process only depends on the gathered information. Every 5 seconds, aircraft in the same sector will simulate action and communicate the action information according to the default order (which can be based on the time when the aircraft enters this sector). With this sector setting, ideally we can reduce the computation time by a factor of 7.

While the sector configuration can help reduce the computation time for the proposed algorithm, it also introduces a new type of conflict called hand-off conflict: when the aircraft is crossing/flying near a boundary of the sector, there is a higher chance for conflicts to happen since the aircraft

does not know the aircraft information from other sectors. To resolve this problem, we introduce one-directional gates on the boundary of sectors, which is denoted as orange rectangles in Figure 4.2 and the direction of the gate is denoted using arrows. The aircraft is only allowed to exit through the available gates and exiting from other locations will be regarded as a LOS state with reward 0, which the algorithm will try to avoid. The width of the gates is designed to be 1,800m, so as to allow two aircraft passing the gates simultaneously. When an aircraft enters a new sector or takes off from a vertiport, it will be assigned an exit gate for this aircraft (if the goal vertiport of this aircraft is not in the current sector) by minimizing the total path:

$$\min\{\sqrt{(p_x - e_x)^2 + (p_y - e_y)^2} + \sqrt{(e_x - g_x)^2 + (e_y - g_y)^2}\} \tag{4.11}$$

where $(p_x, p_y), (e_x, e_y), (g_x, g_y)$ are the position for aircraft, exit gate, and the goal of the aircraft. After assigning the exit gate for this aircraft, the onboard algorithm will guide the aircraft toward its assigned exit gate.

To further reduce the risk of conflicts, the aircraft in one sector can also sense the aircraft information from other sectors that are close to this sector (if the distance between the aircraft and the sector is smaller than 1,500m), but the aircraft does not receive any action information (or flight intention) from them. In summary, each aircraft will receive full information (state and action information) from other aircraft in the same sector, partial information (only state information) from other aircraft that are close to its own sector, and no information from all the remaining aircraft.

4.4.3 Parameter Settings

In the proposed algorithm there are two parameters that can impact the performance of the MCTS algorithm: the number of simulations and the search depth. The number of simulations means the number of roll-outs to simulate during the tree search process. Since there is uncertainty in the environment and aircraft dynamics, more roll-outs will cover more cases and make the algorithm more robust. Search depth means how many steps to look ahead. A more detailed definition can be found in [Yang and Wei (2018)]. Typically for the MCTS algorithm, a larger

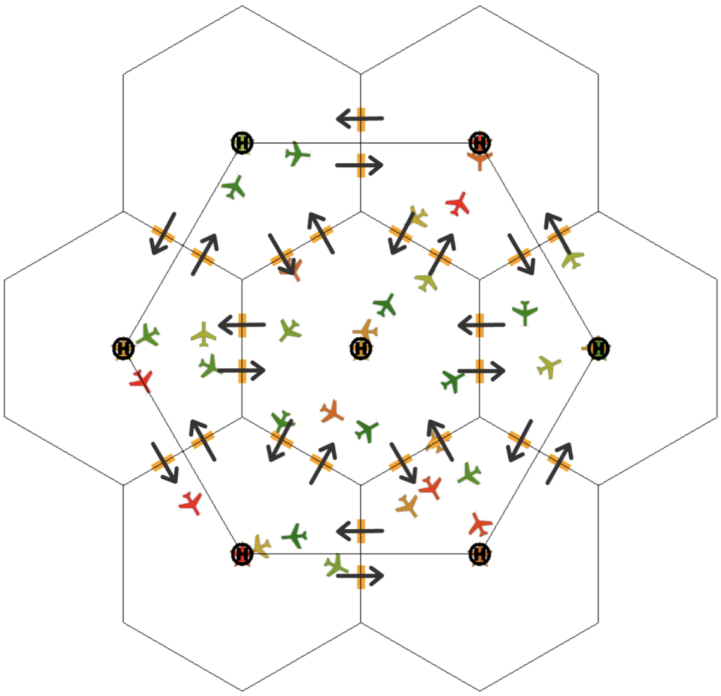


Figure 4.2: The airspace is divided into 7 sectors to reduce the computation time for the proposed algorithm.

search tree (more simulations and deeper search depth) can lead to better performance of the algorithm but need longer computation time. In previous work [Yang and Wei (2018)], it is found that setting the number of simulations to 100 and search depth to 3 in this problem can have decent performance in terms of the number of LOS events/NMACs, average en route flight time, and average onboard computation time. Thus in this dissertation we adopt the same parameter setting. We also noticed that when an aircraft is far from the other aircraft, a smaller tree is enough to find the good action. So to speed up the algorithm, when the distance of an aircraft to its closest aircraft is larger than 2 km, we set the number of simulations to 30 and search depth to 2.

4.4.4 Case Studies and Results

Based on the above simulator setting, we conducted the following three case studies to illustrate the performance of the proposed algorithm.

4.4.4.1 Case study I - On-demand air transportation

In the first case study, based on the airspace configuration we define a demand model that generates flight requests stochastically. At each vertiport, after the taking off of the previous aircraft, the time interval for next aircraft to take off is uniformly distributed between 1 minute and 2 minutes and the newly generated flight request will choose a random vertiport as its destination. Then the simulator will be kept running until 10,000 aircraft have been generated. During the running of this simulator, the number of LOS events and NMACs (short for near mid-air collision) and the average computation time to make each decision will be recorded and compared. Here the NMAC standard is defined to be 500 feet by the Aeronautical Information Manual (7-6-3) [Federal Aviation Administration (2017)].

Then we conducted 5 independent experiments and there are 10,000 aircraft generated in total in each experiment. The code implementation of this algorithm is available on GitHub[1] and a short video demo for this case study can be found on YouTube[2].

Table 4.1 shows the average number of LOS events and NMACs per flight hour and standard error from the simulation results over 5 independent experiments with and without sector configuration (referred to as sectored airspace and unsectored airspace in the following). From this table we see that the LOS event happens around 1×10^{-2} per flight hour and NMAC happens around 1×10^{-4} per flight hour. This table shows the proposed algorithm has promising performance for guidance and separation assurance, and the introduction of the airspace sector help reduce the LOS/NMAC risk for aircraft. Note that the recorded number of NMACs during simulation is in the absence of a collision avoidance system such as TCAS or ACAS-X. A direction of future work would be to integrate our separation assurance model with a collision avoidance system as the final layer of protection. Furthermore, all results shown in this work have not been integrated with any strategic flight plan deconfliction, traffic flow management, or flow control. We expect better safety performance once integrating these components.

[1] https://github.com/xuxiyang1993/Multi_MCTS_Guidance_Separation_Assurance
[2] https://www.youtube.com/watch?v=2cbRUig4G_I

Table 4.1: The simulation results of MCTS algorithm averaged over 5 independent experiments.

	average LOS per flight hour	average NMACs per flight hour
sectored airspace	$(1.05 \pm 0.34) \times 10^{-2}$	$(1.45 \pm 2.90) \times 10^{-4}$
unsectored airspace	$(1.31 \pm 0.22) \times 10^{-2}$	$(2.91 \pm 3.57) \times 10^{-4}$

Figure 4.3 plots the computation time needed for all the aircraft to finish running the onboard algorithm to decide the joint actions. This shows the computation time for both unsectored airspace and sectored airspace are growing with the increase of the number of aircraft, and the sectored airspace greatly reduces the computation time to real-time levels. For the 30 aircraft case, the algorithm with airspace sectorization only takes less than 500 ms to issue the actions for all the aircraft. Note here we record the longest computation time among the 7 sectors. Since most of the time the aircraft are not evenly distributed in the 7 sectors, the computation time does not achieve 7 times faster and the variance among 5 independent experiments is larger than that in unsectored airspace.

Note here that the MCTS algorithm is a statistical anytime algorithm [Browne et al. (2012)]. This means the algorithm can stop running at anytime by returning the current best action, and longer computation time generally leads to better action advisory. This is beneficial for the case when an aircraft needs a maximum acceptable run time.

Although the introduction of airspace sectorization help increase the safety level and reduce the computational time of the proposed algorithm, it also requires metering the aircraft to the exit gate of each sector which will force the vehicles to deviate from their preferred optimal route and result in suboptimal trajectories. Thus in the numerical experiment we also studied the en route flight time in sectored airspace and unsectored airspace. Here we divided the route option into 3 categories based on the distance between the start vertiport and target vertiport. Recall in Figure 4.1 we labeled the vertiports with different ID numbers. In this vertiport map, route 1 is the shortest (e.g., from vertiport 1 to vertiport 2), route 2 has the medium length (e.g., from vertiport 3 to vertiport 7), and route 3 has the longest path (e.g., from vertiport 3 to vertiport 6). The three different routes are plotted in Figure 4.4. Table 4.2 shows the en route flight time of sectored

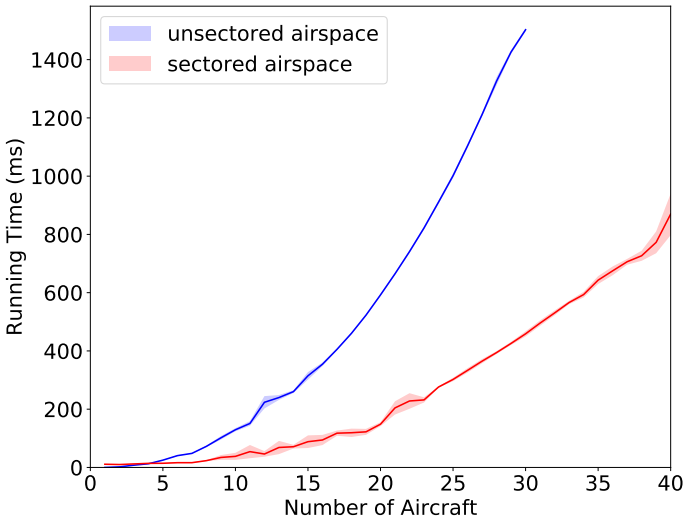


Figure 4.3: The computation time with increasing number of aircraft for the unsectored airspace and sectored airspace.

airspace and unsectored airspace for different route options assuming there is no intruder aircraft. From this table we can see for route 1, 2, and 3, the airspace sectorization increases the flight time by 6s, 20s, and 7s respectively, which is acceptable in trade of safety level and computation time. For route 1 and route 3, the difference is small since the deviated trajectory in sectored airspace is close to the preferred optimal trajectory in unsectored airspace.

Table 4.2: Flight time (en route air time) in seconds for sectored airspace and unsectored airspace.

route	1	2	3
sectored airspace	340.60 ± 7.91	606.56 ± 10.58	685.71 ± 10.81
unsectored airspace	334.46 ± 7.48	586.43 ± 9.88	678.81 ± 10.75

In Figure 4.5, we recorded the number of en route aircraft at each time step and plotted the resulting histogram. In this plot, x-axis is the total number of en route aircraft and y-axis denotes their frequency. From this figure we can see most of the time, there are $25 \sim 40$ aircraft flying in the air, which is approximately $22.55 \sim 36.08$ aircraft per 10,000 km^2.

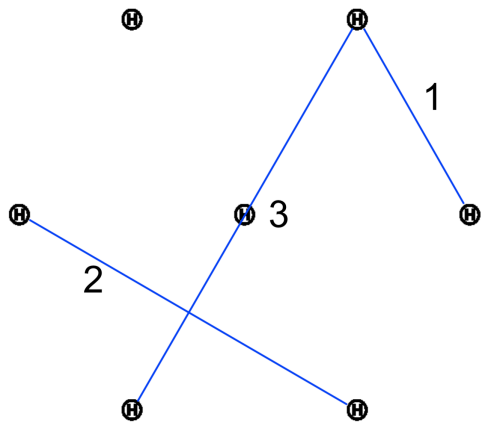


Figure 4.4: Three different routes in the above vertiport setting.

4.4.4.2 Case study II - eVTOL with different priority

In the second case study, we further analyze the en route flight time for the aircraft with priority, since in some cases we may have an emergency flight request which needs to reach its destination in the shortest possible time (e.g., a law enforcement eVTOL, or an ambulance eVTOL). This case can also happen when an aircraft flying in the air has a fault in its electric propulsion system. To let these emergent aircraft arrive goal in a shorter time, in this experiment we divide all the aircraft into two categories: each aircraft is classified into high/low priority, which is used to denote the emergency level for an aircraft. When an aircraft takes off, the priority will be randomly assigned to the aircraft in the numerical experiment. In real-world applications, an eVTOL will be identified and approved with different priorities depending on its trip purpose.

When the algorithm makes decisions for the aircraft, it will first generate actions for all the aircraft with high priority, by only considering the high priority aircraft information. Then it will next generate actions for all the remaining aircraft, given the actions made for the high priority aircraft.

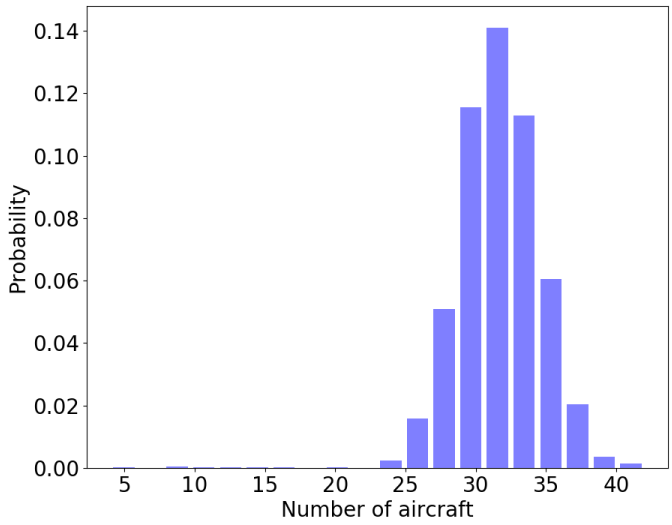


Figure 4.5: The histogram for the total number of en route aircraft.

Specifically, suppose in the current sector we have n aircraft, out of which p aircraft are with high priority. Then the algorithm will first initialize the actions for these p aircraft:

$$\mathbf{a}_{\text{high}} = \{a_1, a_2, \cdots, a_p\} \tag{4.12}$$

where a_i is the action for aircraft i and $a_1 = a_2 = \cdots = a_p = 0°/s$. The algorithm will generate actions based on the high priority aircraft information $[s_1, \cdots, s_p]$ where s_i denotes the state information for aircraft i. After generating the optimal actions for these p high priority aircraft we get

$$\mathbf{a}^*_{\text{high}} = \{a^*_1, a^*_2, \cdots, a^*_p\} \tag{4.13}$$

The algorithm will then next initialize the joint action for all the remaining aircraft as

$$\mathbf{a}_{\text{low}} = \{a^*_1, \cdots, a^*_p, a_{p+1}, \cdots, a_n\} \tag{4.14}$$

where $a_{p+1} = \cdots = a_n = 0°/s$. Then the algorithm will generate actions for these aircraft with all of the aircraft information $[s_1, \cdots, s_n]$.

In this experiment, we compare the average en route flight time of aircraft with different priorities for the 3 different routes, which is recorded in Table 4.3. In this table we also added the

optimal en route flight time where there is no intruder aircraft, from which we can see the flight time of high priority aircraft is closer to the optimal path flight time comparing with low priority aircraft, and high priority aircraft can save 14s, 34s, 43s en route time for each route comparing with low priority aircraft.

Table 4.3: Flight time (en route air time) in seconds for high priority and low priority aircraft.

route	1	2	3
clear path	340.60 ± 7.91	606.56 ± 10.58	685.71 ± 10.81
high priority	344.93 ± 0.32	619.98 ± 0.70	703.14 ± 1.31
low priority	358.80 ± 0.83	653.38 ± 0.49	746.51 ± 2.32

4.4.4.3 Case study III - Roundabout stress test

In this case study, the proposed algorithm is evaluated against a stress-test set of multi-threat scenarios randomly generated from an encounter model, similar to the experiments presented in [Ong and Kochenderfer (2016)]. This case study is helpful since a large part of current commercial interest surrounding UAM stems from the potential to accommodate a large number of eVTOL aircraft and it is estimated there will be 23,000 aircraft flying major routes within the UAM network by 2035 [Porsche Consulting study (2018)].

In each encounter scenario, the number of aircraft in the multi-threat encounters is ranging from 10 to 20, distributing uniformly on an annulus. Specifically, the annulus had inner and outer radii of 10km and 15km, and if a new aircraft added is closer to other aircraft than 2km, we resample the new aircraft position to avoid initializing aircraft already in the LOS state. The goal position of each aircraft is set to be the symmetric point in the annulus with respect to this aircraft's position, so the headings of each aircraft are initialized to point straight towards the annulus center to ensure that all aircraft would have potential conflicts. Figure 4.6 shows an example stress test scenario with 10 aircraft where the positions were initialized uniformly in the annulus.

Figure 4.8 illustrates the performance of the proposed computational guidance algorithm in the stress test scenarios as the number of aircraft increases, where each point denotes the average result

in 1,000 independent encounter scenarios. It plots the probability that an aircraft has LOS/NMAC with another aircraft in one scenario. From Figure 4.7a we can see for the MCTS algorithm the LOS event probability is less than 1% and the NMAC probability is less than 0.1% for each aircraft. We also show the result of the baseline where no actions are taken for all of the aircraft (e.g., all of the aircraft are flying straight towards their respective goals), and plot the average result over 1000 independent experiments. To ensure the aircraft can arrive at their respective goals, we remove the uncertainty for the change of heading angle and only keep the uncertainty of speed. From Figure 4.7b we can see when no actions are taken, over 90% of the aircraft will have at least one LOS event and over 40% of the aircraft will have NMAC with other aircraft in each stress test encounter scenario. The comparison of the two plots in Figure 4.8 shows the promising performance of the proposed computational guidance algorithm even with high-density air traffic.

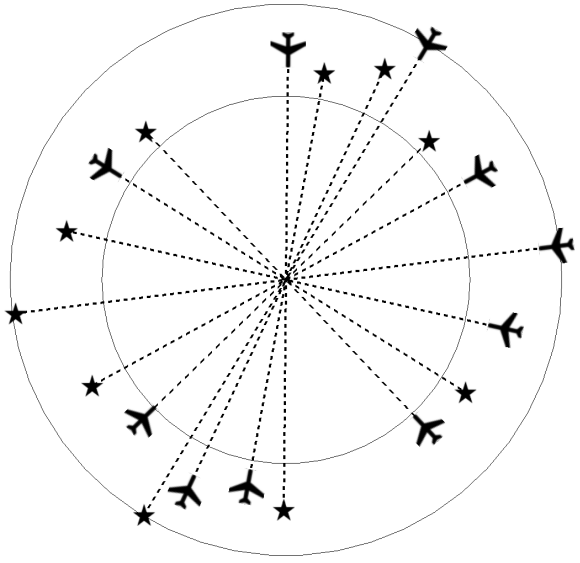
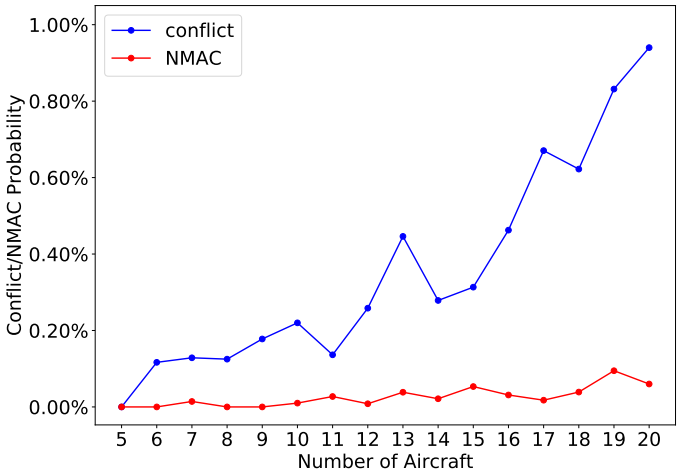


Figure 4.6: One randomly generated stress test scenario. Each aircraft and its destination are connected through dashed line.

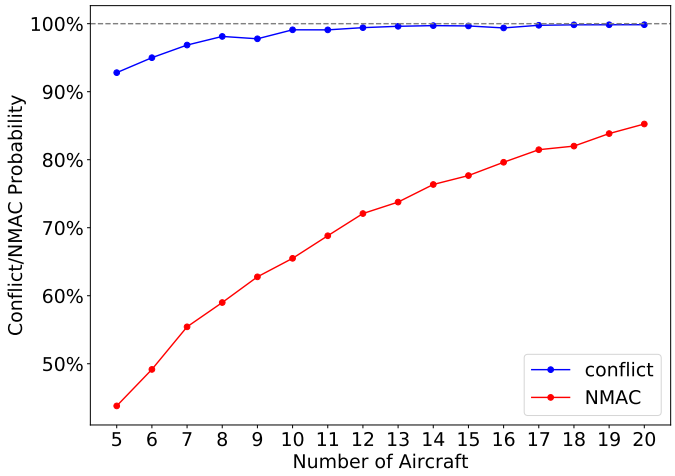
4.5 Conclusion

A message-passing decentralized computational guidance algorithm with a separation assurance capability for multiple cooperative aircraft in urban air mobility is proposed in this chapter. The problem is formulated as a Multiagent Markov Decision Process (MMDP) and then solved by Monte Carlo Tree Search (MCTS) algorithm that can run onboard. A message-passing based coordination mechanism was designed to manage multiple cooperative aircraft. The airspace sectorization is introduced to help to achieve better scalability and safety. Numerical experiments over three case studies show that this proposed algorithm has promising performance to help the aircraft reach their destinations and avoid potential LOS events among them even for the high-density air traffic case.

The proposed concept of operations and algorithm provide a potential solution for decentralized separation assurance to enable safe, efficient, and scalable flight operations in on-demand urban air transportation with high-density air traffic. We found this framework might be a potential solution for certain airspace such as rural or suburban areas. However, the proposed framework is still in the exploratory phase with some limitations. To make the proposed algorithm more practical in real-world applications, the future work should include (1) expanding the action space to altitude changes and speed changes; (2) adapting the proposed algorithm in restricted airspace or structured airspace; (3) incorporating a high-fidelity aircraft dynamics model with more realistic mission profile; and (4) integrating other layers of protection into this framework such as strategic flight planning, flow control, and collision avoidance systems.



(a) MCTS algorithm



(b) Baseline

Figure 4.8: LOS event/NMAC probability as the number of aircraft increases.

CHAPTER 5. MULTI-AGENT AUTONOMOUS OPERATIONS IN URBAN AIR MOBILITY WITH COMMUNICATION CONSTRAINTS

5.1 Introduction

In chapter 3 and chapter 4, we have presented computational guidance algorithms for single aircraft and multiple cooperative aircraft by assuming the perfect communications for vehicle-to-vehicle and ground-to-vehicle channels. However, in real-world applications, air-to-air and air-to-ground communications will be affected by many factors [Rappaport et al. (1996); Orjih (2006)] including bandwidth constraint and communication loss. In this chapter we consider these communication constraints in the design of computational guidance algorithms to achieve multi-agent autonomous, safe and efficient UAM operations.

Specifically, our goal in this chapter is to (1) improve our existing computational guidance algorithms given communication constraints; (2) design air-to-air and air-to-ground communication frameworks to facilitate the computational guidance algorithm; and (3) integrate the decision making and communication mechanisms to guide all the aircraft to their respective destinations while avoiding potential conflicts between them.

In this chapter, we first describe the practical consideration for communication latency, loss, and bandwidth, as well as their impact on the previously proposed computational guidance algorithms. Section 5.3 presents the centralized and decentralized communication frameworks. The numerical experiments and results are described in section 5.4, and section 5.5 concludes this chapter.

5.2 Communication Constraints

When the algorithm is running to issue actions for all the aircraft, the aircraft needs to communicate information between each other (position, velocity, destination location, and intended action for next time step). In our previous work [Yang et al. (2019)], we assume all the aircraft

can communicate perfectly. However, in real-world applications, there are many restrictions for the communication between aircraft [Rappaport et al. (1996)].

5.2.1 Communication Latency

Delay is one of the major concerns in communication systems, which consists of the propagation time (limited by light speed) and processing latency (e.g., the computational time overhead of demodulation and decoding). In typical wireless communication systems, the delay in point-to-point communications could be in the order of milliseconds. A significant communication delay could hinder the timely delivery of information and thus cause performance degradation in real-time control systems.

In this chapter, we use previous work on aircraft wireless communication systems [Orjih (2006)] to decide the communication time between two aircraft or between ground-based centralized controller and aircraft. According to [Orjih (2006)], the onboard communication system can provide a full-duplex 2 Mbps/256 kbps (downlink/uplink) connection to a fixed ground station via satellite, where the uplink process will limit the communication time. In the proposed algorithm, the aircraft needs to send its own state information including position, velocity, destination, and intended action, and the size of this information is smaller than 256 bits (each number takes 32 bits of memory). Thus the communication can finish under 1ms, which is negligible comparing with the computation time for the online guidance MCTS algorithm.

5.2.2 Communication Bandwidth

The data transmission rate is limited by the available bandwidth, namely the frequency band used for the communications. The broader, the faster. Usually bandwidth is expensive due to the congestion in the frequency bandwidth allocation, especially in the proposed computational guidance algorithm, where the information broadcast and communication will be used extensively. Therefore, there is a tradeoff between the bandwidth resource and cost, which means the aircraft needs to try decreasing the communication frequency and only sends the necessary information.

Usually an onboard communication device cannot transmit and receive in the same band at the same time due to high self-interference (the power of transmitted signals is orders of magnitude higher than that of the received signal). Besides, since the broadcast nature of wireless communications, different communications links in a wireless communication network cannot transmit simultaneously [Li et al. (2008)].

In the proposed algorithm, the communication between two aircraft includes two processes: (1) one aircraft receives information from all other aircraft; (2) one aircraft transmits the action information to all other aircraft. Due to the factors described above, in this dissertation we assume aircraft receives/transmits information from/to other aircraft one after another.

5.2.3 Communication Loss

During the aircraft en route flight, communication loss between two aircraft or between aircraft and the ground-based centralized controller may happen [SKYbrary (2019a)], resulting from:

1. Communication equipment problems caused by the malfunction or complete failure of aircraft or ground equipment (becoming less of an issue with improved system redundancy);

2. Radio interference where transmissions other than those from authorized users of an RTF frequency interfere with radio reception;

3. Blocked transmissions due to the tall buildings.

When a communication loss happens, the centralized controller will not be able to receive the state information from the aircraft lost communication, which will cause a higher probability of conflict for this aircraft when the centralized controller is making decisions for all of the aircraft. Besides, the centralized controller is not able to send action information to some aircraft, which may cause the aircraft to take action resulting in conflicts with other aircraft or even an unauthorized entry of designated airspace. This may lead to disruption of air traffic, causing risk to other airspace users and increased workload for pilots and controllers.

5.3 Solution Method

In our previous work [Yang et al. (2019)], we proposed a coordination mechanism design, where at each decision making step, we set a decision sequence for all aircraft and let them make decisions one after another. After one aircraft selects the action, it will broadcast this action information to all the other aircraft, the aircraft making decisions later can utilize this information to select better action. During this process, the state information and action information of each aircraft need to be synchronized/communicated.

In this dissertation, we consider two communication strategies. In the first case, at each decision time step, all of the aircraft send their own information (position, velocity, destination) to a ground-based centralized controller which is located at the center of the airspace. Then the centralized controller will run the proposed algorithm and send the result joint action to each aircraft. In the second case we assume the centralized controller is down and the aircraft needs to use their onboard computer to run the proposed algorithm. In this case the aircraft will communicate the state information and joint action information between each other. We describe the two cases in the following two subsections.

5.3.1 Framework 1: Centralized Control and Air-to-ground Communications

In this case, a centralized controller is responsible to receive information from all of the aircraft and issue the result joint action to each aircraft. In this process, communication loss may happen and we assume the loss is two-directional (the aircraft cannot send information to the centralized controller and the centralized controller cannot issue action to the aircraft). Whenever a communication loss happens, the centralized controller will lose the state information of some en route aircraft. In our work the centralized controller will use the state and action information from the

last step to predict the current position/velocity of the aircraft:

$$\begin{aligned} v &= v^- \\ \psi &= \psi^- + a^- \\ x &= x^- + v\cos\psi \\ y &= y^- + v\sin\psi \end{aligned} \tag{5.1}$$

where the "super minus" sign denotes the state and action information from last decision step. Note here the state prediction from last step information is not accurate since there is uncertainty in the aircraft dynamics (described in Section II.B.), where the noises for heading angle and speed are normally distributed with standard deviation 2° and $5m/s$. As shown in Figure 5.1, the orange point is the aircraft position from last step, the red point denotes the aircraft position for current step without any uncertainty, and the blue points describe the distribution of current step aircraft position given the uncertainty described above.

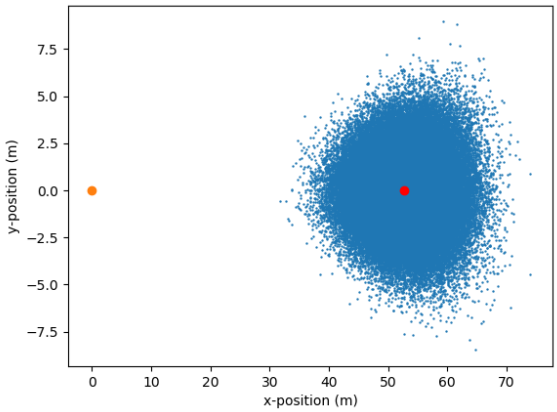


Figure 5.1: The simulated distribution of the aircraft position in the current step is shown in blue as point cloud. Current aircraft position without uncertainty is shown in red inside the blue point cloud, and the orange point on the left is the aircraft position from the previous step.

To ensure the safe separation in scenarios where the centralized controller has to predict the current state of the aircraft that lost communication, the strategy is to increase the separation distance to mitigate the prediction error. More specifically, we use the property of normal distribution

to decide the increased separation distance. Since we have

$$P(-2.33\sigma_\psi < \psi < 2.33\sigma_\psi) = 99\% \tag{5.2}$$

$$P(-2.33\sigma_v < v < 2.33\sigma_v) = 99\% \tag{5.3}$$

where $\sigma_\psi = 2^\circ$ and $\sigma_v = 5m/s$. If we select a point from the above range, then we can guarantee it covers 98% of the position points. Since it can be calculated that the furthest point in the above range from the red point is 12 m, we will increase the separation distance by 12 m for each time step for our prediction while the communication is lost for this aircraft.

With the increased separation distance for the aircraft lost information and by assuming it fly straight, the centralized controller will issue actions for all of the other aircraft by following the process described in previous work [Yang et al. (2019)].

5.3.2 Framework 2: Decentralized Control and Air-to-air Communications

In centralized control scenario, we found some cases where two aircraft close to each other lost communication to the centralized controller simultaneously. In this case, these two aircraft have a higher risk for conflict since the centralized controller is not able to issue actions to these two aircraft. Thus in this case study we propose an algorithm to allow aircraft-to-aircraft communication for decentralized conflict resolution, where the aircraft will send its state/action information to other aircraft for cooperative conflict resolution purposes. Since in real world, aircraft close to each other are very unlikely to have communication loss, thus they can avoid potential conflicts through communication and coordination.

In the following, we will discuss the algorithm running and communication process with/without communication loss, as shown in Figure 5.3. At first, assume currently we have three aircraft flying in the airspace without any communication loss (the arrow means they can send information to each other freely), as shown in Figure 5.2a. To run the decentralized MCTS algorithm, we first need to maintain an order among the aircraft (which is decided according to their order to enter the airspace), then the aircraft can synchronize joint action information according to the maintained order. At the beginning of the process, all of the aircraft (aircraft 2 and aircraft 3 in this case) will

send its own information (position, velocity, destination) sequentially according to the maintained order to aircraft 1, which is denoted as the receive block for aircraft 1. Since we have two aircraft flying in the air, the receiving process can be finished in 2 ms, and this number will increase with the number of the aircraft. With all these aircraft information, aircraft 1 can now begin to make its decision by building a MCTS search tree. After aircraft 1 makes a decision and selects an action, it will broadcast this action information to all of the other aircraft sequentially in the same order as the receiving process. As aircraft 1 is making its decision, aircraft 2 begins to receive information from other aircraft. Although aircraft 2 finishes receiving the information from other aircraft in a shorter time compared to aircraft 1's decision process, it cannot start running the MCTS algorithm until it receives action information from aircraft 1, which is necessary information for aircraft 2 to run the onboard MCTS algorithm. Then the same process follows for aircraft 3. After all of the aircraft updates its own action, the algorithm running and communication process finishes and all of the aircraft will take the action according to their own MCTS algorithm result.

It can also happen two aircraft have communication loss and cannot send state/action information to each other, as shown in Figure 5.2b, where aircraft 2 and aircraft 3 cannot send information to each other. In this case, the algorithm running and aircraft communication process will be the same as the case in Figure 5.2a for aircraft 1 and aircraft 2. Starting from aircraft 3, when the receiving process of aircraft 3 finishes, it will be aware of the communication loss between itself and aircraft 2 since it did not get aircraft 2's information. Then, it can immediately proceed to run the onboard MCTS algorithm without state/action information from aircraft 2. Since communication loss usually happens between aircraft with a large distance, it is safe to run the MCTS algorithm without knowing position/velocity information and action/intent information from aircraft 2. Besides this difference, all the remaining process stays the same with no communication loss case.

5.4 Numerical Experiments and Results

5.4.1 Simulator

To validate the performance of the proposed algorithm, we use the free flight simulator, UAM network, and demand model described in section 4.4.

In the centralized scenario, the communication loss happens when the aircraft is 3km away from any vertiport. At each time step, there is 5% probability that a communication loss may happen, where the aircraft and the centralized controller cannot communicate information between each other. Whenever a communication loss happened, a uniformly distributed random variable in the range $[2, 5]$ will be generated to represent the communication loss time interval. During the communication loss, the aircraft will take the action to fly straight if it does not receive action information from the centralized controller and the centralized controller will use last step information to predict the aircraft state information at the current time step.

In the decentralized scenario, if two aircraft are at least 6km away from each other, they could have communication loss. The probability of communication loss increases from 0 to 0.5 linearly as the distance between them increases:

$$P(\text{communication loss}) = \text{clip}(\frac{d}{60000} - 1, 0, 0.5) \tag{5.4}$$

where d denotes the distance between two aircraft. Then the aircraft will make decisions according to the process described in Section IV.

The simulator will be kept running until 10,000 aircraft have been generated. During the running of this simulator, the number of conflicts and NMACs (short for near mid-air collision), the total number of aircraft generated, the number of aircraft which reached goals, and the average computation time to make each decision will be recorded and compared. The near mid-air collision (NMAC) standard is defined to be 500 feet by the Aeronautical Information Manual (7-6-3) [Federal Aviation Administration (2017)].

5.4.2 Results

The numerical experiment described above is conducted over 5 random seeds and in each experiment, there are 10,000 aircraft generated in total. The code implementation of this algorithm is available on GitHub[1]. The result of the numerical experiment is shown below.

Table 5.1 and Table 5.2 show the number of conflicts/NMACs and the number of aircraft reached the goal for centralized control and decentralized control scenarios. In the centralized case, we can see for all the 10,000 aircraft generated, the conflict probability is around 0.23% and the NMAC probability is 0.024%. For the decentralized scenario, the conflict probability is about 0.3% and NMAC probability is 0.006%.

The result shows the proposed algorithm is pretty efficient at ensuring safety separation between aircraft. Comparing with previous work [Yang et al. (2019)] where the conflict/NMAC probability is 0.2% and 0.004%, the performance of the proposed algorithm for centralized control and decentralized control is a little worse since here we considered the communication constraints. Besides, comparing with centralized control, decentralized control is more robust since the aircraft can communicate with each other. This shows that a communication network with more links is beneficial to the whole guidance and collision avoidance system.

Table 5.1: Performance of the algorithm for centralized control.

	mean	variance	min	max
Number of Conflicts	23	44.8	15	35
Number of NMACs	2.4	2.64	0	5
Number of Aircraft Reached Goal	9995.2	10.56	9990	10000

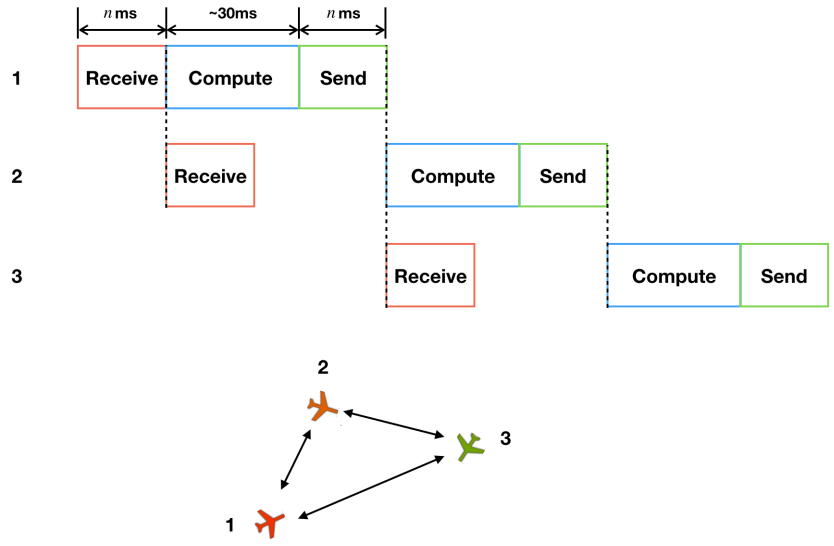
Table 5.2: Performance of the algorithm for decentralized control.

	mean	variance	min	max
Number of Conflicts	30.2	27.8	25	37
Number of NMACs	0.6	0.24	0	1
Number of Aircraft Reached Goal	9999.8	0.96	9998	10000

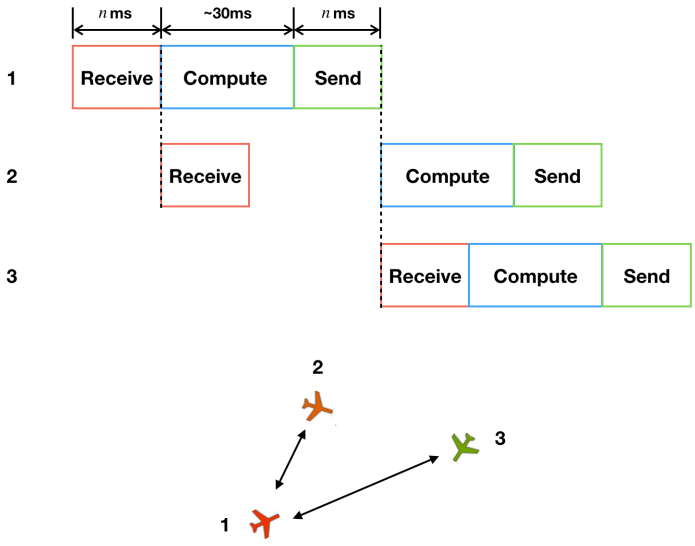
[1] https://github.com/xuxiyang1993/MultiAgent-Guidance-Communication

5.5 Conclusion

Building upon the computational guidance algorithms proposed in previous chapters, in this chapter we consider the communication constraints and build communication frameworks to make the proposed algorithms more practical. In centralized control, we use forward propagation for the centralized controller to predict the state information of aircraft that lost communication. In decentralized control, we design an efficient communication framework for aircraft-to-aircraft communication. With the communication constraints including latency, bandwidth, and loss, we propose a robust computational guidance algorithm for multiple cooperative eVTOL aircraft in urban air mobility. Numerical experiments show that this proposed algorithm has promising performance to help an aircraft reach its destination and avoid potential conflicts with other aircraft.



(a) Without communication loss



(b) With communication loss

Figure 5.3: The algorithm running and communication process for decentralized control.

CHAPTER 6. LEARNING-BASED PERCEPTION FOR ONBOARD MULTIPLE-OBJECT DETECTION AND TRACKING

6.1 Introduction

In chapters 3, 4, and 5 we presented autonomous computational guidance systems for the single aircraft case and multiple cooperative aircraft case with communication constraints, where we assumed we have perfect sensing/perception technologies. Another key component for the safe autonomous operations for aircraft is an effective onboard perception system that supports sense-and-avoid functions in previous chapters. To improve the safety of autonomous flight operations, in this chapter we investigate learning-based perception models. This work is motivated by many anticipated benefits for both online (real-time) and offline detection in aircraft-video feeds.

For offline uses, let us consider the difficult process of obtaining a FAA waiver for a beyond-visual-line-of-sight mission. In order to make the case that a proposed mission is safe, the applicant must quantify the risk to people on the ground and in moving vehicles in the mission area. One way to do this is to survey the underlying ground area with a series of safe visual line of sight drone missions. We can then identify pedestrians and moving cars in the video footage from each flight using our proposed object detection models. By fusing the video data with flight telemetry (latitude, longitude, altitude) and the camera pose (yaw, pitch, and roll), we can approximate the density of humans at risk in the area on the ground under the drone mission during a specific time period. We can then suggest safer routes that avoid areas of higher risk such as schools or playgrounds.

The online use cases are more tactical and safety-critical, where our computer vision models can support real-time decision making from onboard autonomy or from a remote pilot. With a real-time pedestrian and vehicle detection capability, we could significantly enhance safety in a populated, dynamic mission area during the landing phase of a regular package delivery, or an emergency

landing. We could also use the perception models to autonomously avoid walking pedestrians and moving cars and locate a safe landing space. Additionally, we could create real-time safety tools that alert the pilot when the UAS is approaching a higher risk area with many pedestrians or cars.

In this chapter, we first describe the learning-based perception model. In section 6.3 we describe the training process of the learning-based model on a publicly available dataset. Section 6.4 presents the results of the trained perception model and section 6.5 concludes this chapter.

6.2 Background of Learning-based Computer Vision: RetinaNet

6.2.1 Class Imbalance Problem of One-Stage Detector

In two-stage detectors the region proposal model at the first stage will narrow the number of candidate object locations to a small number (e.g., 1,000 or 2,000), while filtering out most background samples. At the second stage, classification is performed for each candidate object location. Sampling heuristics using a fixed foreground-to-background ratio (1:3), or online hard example mining (OHEM) [Shrivastava et al. (2016)] to select a small set of anchors (e.g., 256) for each mini-batch. Thus, there is manageable class balance between foreground and background [Lin et al. (2017b)].

For one-stage detectors, a much larger set of candidate object locations is regularly sampled across an image (∼100k locations), which densely cover spatial positions, scales and aspect ratios. The training procedure is still dominated by easily classified background examples. To resolve this class imbalance problem, a new loss function called Focal Loss [Lin et al. (2017b)] was proposed which is a more effective alternative to previous approaches, which has the following form

$$FL(p_t) = -(1 - p_t)^{\gamma} \log p_t \tag{6.1}$$

where p_t is the model's estimated probability and γ is the tuneable focusing parameter.

As shown in Figure 6.1, when γ is set to a positive number, the easy well-classified examples (background) will contribute less to the total loss function and the hard misclassified examples

(foreground objects) will contribute more to the total loss function, thus making the optimization algorithm concentrate more on the hard examples.

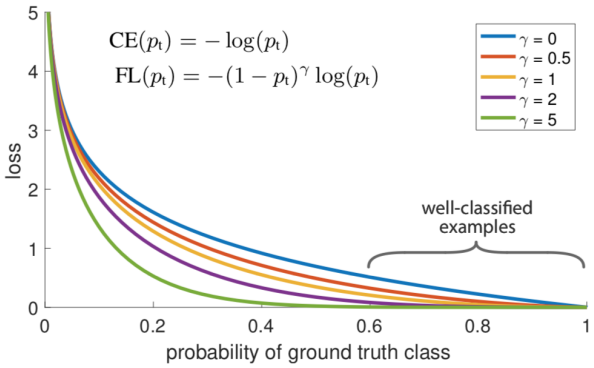


Figure 6.1: Focal Loss adds a factor $(1 - p_t)^\gamma$ to the standard cross entropy loss function. Setting $\gamma > 0$ reduces the relative loss for well-classified examples, putting more focus on hard, misclassified examples [Lin et al. (2017b)].

6.2.2 Retina Detector Architecture

RetinaNet is a single unified network consisting of two parts: a 50-layer ResNet [He et al. (2016)] backbone network which is used for deep feature extraction and two task-specific subnetworks which perform convolutional object classification and convolutional bounding box regression. Also, Feature Pyramid Network (FPN) [Lin et al. (2017a)] is used on the top of ResNet for constructing a rich multi-scale feature pyramid from one single resolution input image. More specifically, the FPN layer is built on top of the ResNet architecture [He et al. (2016)], from which pyramid levels P_3 through P_7 are constructed where l indicated pyramid level (P_l has resolution 2^l lower than the input).

In real-world object detection, objects from the same class may be in a wide range of scales in images. A traditional convolution neural network is not good at detecting objects on a small scale since feature maps from higher levels are spatially coarser. By combining higher level features and low level features, FPN can help detect objects in various scales, which is helpful in our case since pedestrians in the aerial images are usually of small size.

6.2.3 Anchor Parameter

One of the most important design considerations in the one-stage detector is how densely it covers the space of possible image boxes. Since one-stage detectors use a fixed sampling grid, one popular approach for achieving high coverage of boxes for various sizes (scales and aspect ratios) of objects is to use multiple "anchors" [Ren et al. (2015)] at each spatial position. In the RetinaNet [Lin et al. (2017b)] algorithm, the authors use 9 anchors per location spanning 3 scales ($2^{0/3}, 2^{1/3}, 2^{2/3}$) and 3 aspect ratios [1:2, 1:1, 2:1] for the bounding boxes, and the anchor sizes are of 32, 64, 128, 256, 512, according to the 5 different pyramid levels. While this anchor configuration has decent performance on the COCO dataset [Lin et al. (2014)], it is not suitable for the images taken from drones (e.g., in the dataset we are using in this chapter, pedestrians are usually smaller than 32×32 pixel, which is the size of the smallest anchors). In this dissertation, we drop the biggest one of 512 and instead add a small anchor of size 16, which can cover the scale of 16 - 407 pixels with respect to the network's input image.

6.3 Experiments

We present experimental results on the *VisDrone2019-Det* benchmark dataset [Zhu et al. (2018)]. We use a Linux workstation running Ubuntu 18.04 with an Intel(R) Xeon(R) E5-1650 v2 CPU @ 4.00GHz (12 CPUs), 256GB RAM, and one NVIDIA Titan Xp GPU (12GB) to train and evaluate the models in our experiments.

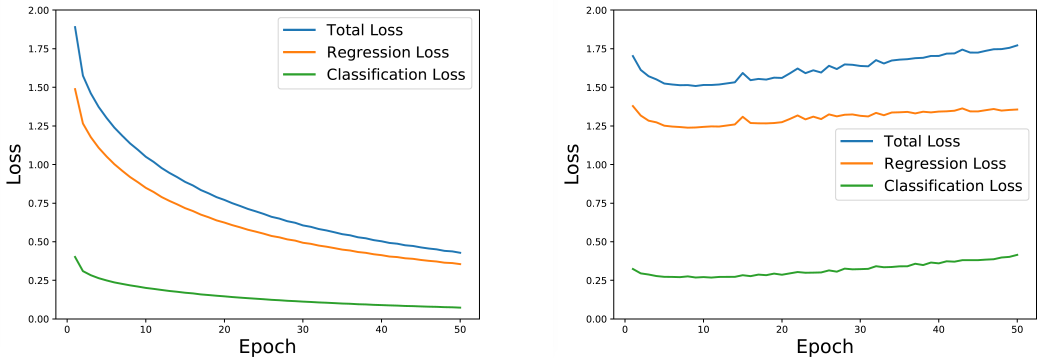
6.3.1 Dataset

VisDrone2019-Det dataset consists of 7,019 static images captured by various drone-mounted cameras, covering a wide range of aspects including location (taken from 14 different cities separated by thousands of kilometers in China), environment (urban and country), objects (pedestrian, vehicles, bicycles, etc.), and density (sparse and crowded scenes). Also, the dataset was collected using various drone platforms (i.e., drones with different models), in different scenarios, and under various weather and lighting conditions. These frames are manually annotated with more than 2.6

million bounding boxes of 10 target objects (i.e., pedestrian, person, car, van, bus, truck, motor, bicycle, awning-tricycle, and tricycle). The RetinaNet model is trained on the training set and evaluated on the validation set.

6.3.2 Model Training

We did experiment using RetinaNet with ResNet-50-FPN backbones. The base RetinaNet is pre-trained on the COCO dataset [Lin et al. (2014)], then trained using the adam algorithm with learning rate 10^{-5}. We use horizontal image flipping as the only form of data augmentation. The training loss is smooth L_1 loss for box regression [Girshick (2015)] and focal loss for the object classification [Lin et al. (2017b)]. Figure 6.3 shows the loss during training, from which we can see the model begins to overfit the training data after 20 epochs. Figure 6.4 plots the weighted mAP of the 10 classes on the validation dataset. At epoch 17, the RetinaNet algorithm achieves the highest mAP of 30.21% on the validation set.



(a) training dataset (b) validation dataset

Figure 6.3: The loss during training on training dataset and validation dataset.

VisDrone2019-Det is a very challenging dataset with high category imbalance. As shown in Table 6.1, the category with a large number of instances dominates the loss function, hence this category obtains a higher mAP score (e.g., car and pedestrian). The mAP for the car is higher than the pedestrian is because the relative scale of the car is larger and hence easier for the algorithm to

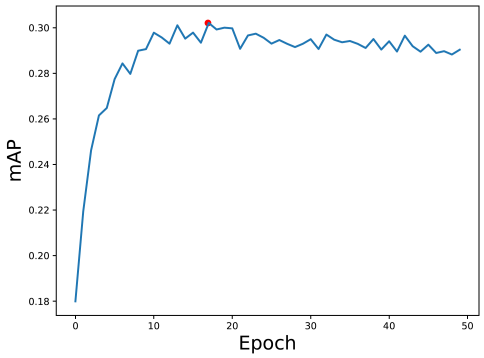


Figure 6.4: The mAP score on the validation dataset during training.

detect. In future work, we are planning to use data augmentation and some oversampling techniques to deal with the class imbalance issue.

Table 6.1: Detection performance for each category on validation set.

Class	Images	Instances	mAP
pedestrian	548	8844	36.78
people	548	5125	27.33
bicycle	548	1287	11.22
car	548	14064	71.30
van	548	1975	30.37
truck	548	750	27.71
tricycle	548	1045	16.56
awning-tricycle	548	532	6.88
bus	548	251	39.09
motor	548	4886	34.82
overall	548	38759	30.21

6.4 Results

We test the performance on images and videos taken by our drone using the model trained above with the highest mAP. All of our drone images and videos were captured with a DJI Mavic Pro 2 drone. The Mavic Pro 2 can shoot video at 4k resolution but it is stored on a memory card on the drone. These videos can be retrieved from the drone after the flight and analyzed

offline. The Mavic Pro 2 can also live-stream video at 1080p resolution. For testing in real time, we used DJI's Microsoft Windows SDK to stream the drone video to our models in real time on our GPU-equipped laptop. To use this in the field, we needed to bring the laptop with us. Ideally, we would like to be able to use a smartphone for object detection, but they do not have the computing power to recognize objects in real-time with our model. We are working on speeding up inference to the point where this is possible.

Figure 6.6 shows two sample detection results of the trained model for the images taken by our drone, where the blue rectangle represents detected pedestrian and the orange rectangle denotes detected cars. From this figure we can see if the angle of the camera is closer to bird eye view, the performance will be worse since the dataset the model trained on doesn't contain many bird eye view images. A sample detected video is also available on YouTube[1].

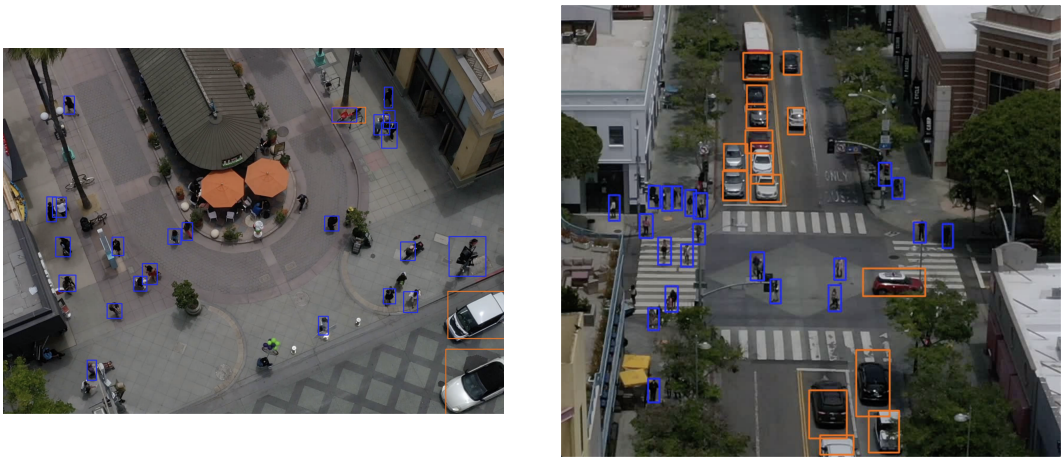


Figure 6.6: Visualized detection results of RetinaNet trained model.

6.5 Conclusion

In this chapter, we propose an efficient learning-based perception algorithm that can be deployed on a drone. By tuning the network architecture and anchor parameter, the proposed algorithm can be used to detect small pedestrians and cars from aerial images and videos. After training the model

[1] https://www.youtube.com/watch?v=XFyoZfURRlM

on a drone image dataset, the proposed algorithm achieves high mAP for detecting pedestrians and cars. Real-world cases also show this algorithm is promising for detecting pedestrians and cars, which can help the drone estimate the risk level through the detection results.

The main contribution of this work is the first deep-learning based computer vision model for pedestrian detection and tracking. This work is more challenging than street-level pedestrian detection algorithms used by self-driving cars as we are dealing with a variety of camera angles (pitch, in particular), as well as heights. That is, pedestrians and cars need to be detected from a greater distance to the UAV than from a self-driving car, and viewed from above, they often appear in a very small part of a high resolution image. In contrast with existing research with ground-level pedestrian detection, the developed algorithm achieves highly accurate multiple pedestrian detection from a bird-eye view, when both the targets and the aircraft platform are moving.

CHAPTER 7. SAFETY VERIFICATION WITH ADAPTIVE STRESS TESTING FOR UAS TRAFFIC MANAGEMENT DECISION MAKING SYSTEMS

7.1 Introduction

In chapters 3, 4, and 5, we propose real-time decision-making systems for autonomous flight operations to accommodate the large-scale operations and complex environments in UAM. However, the proposed decision-making systems will require validation and verification processes to ensure that they meet the appropriate safety and reliability criteria. Verifying the safety of autonomous systems can be challenging, because these systems are inherently complex internally, and interact with external systems and environments in non-trivial ways.

To validate and verify the given autonomous systems efficiently, in this chapter we propose a new methodology for finding the failure modes for the given decision-making systems. The contributions of this work include: (1) proposing a novel and sample efficient approach for adaptive stress testing that uses Bayesian optimization, (2) expanding on the approach in a way that enables the methodology to also optimize the system with respect to the discovered failure modes, (3) applying this approach to validate and optimize strategic decision-making services for the UAM and the package delivery uses cases.

In this chapter, we first briefly review the background of Bayesian Optimization in section 7.2. In section 7.3, we formulate the validation problem as an optimization problem. Section 7.4 presents the result for failure scenario identification and path planner optimization. Section 7.5 concludes this chapter and describes future work.

7.2 Bayesian Optimization with Gaussian Process Priors

Bayesian optimization is a derivative-free sequential design strategy for global optimization of black-box functions [Mockus (2012)], which has been shown to outperform other state of the art global optimization algorithms on a number of challenging optimization benchmark functions [Jones (2001); Snoek et al. (2012)]. For continuous functions, Bayesian optimization typically works by assuming the unknown function was sampled from a Gaussian process (prior) and maintains a posterior distribution for this function as observations are made sequentially. We will briefly overview Gaussian Processes and Bayesian Optimization in the next subsections.

7.2.1 Gaussian Processes

Gaussian Process (GP) provides a convenient and powerful class of non-parametric statistical models over function spaces. Specifically, a Gaussian process is a stochastic process such that any finite subcollection of random variables has a multivariate Gaussian distribution [Rasmussen (2003)]. In particular, a collection of random variables $\{f(x) : x \in \mathcal{X}\}$ is said to be drawn from a Gaussian process with mean function $m(\cdot)$ and covariance function $k(\cdot, \cdot)$ if for any finite set of elements $x_1, ..., x_m \in \mathcal{X}$, the associated finite set of random variables $f(x_1), ..., f(x_m)$ have distribution

$$\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_m) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(x_1) \\ \vdots \\ m(x_m) \end{bmatrix}, \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_m) \\ \vdots & \ddots & \vdots \\ k(x_m, x_1) & \cdots & k(x_m, x_m) \end{bmatrix} \right) \tag{7.1}$$

we denote this using the notation

$$f(\cdot) \sim \mathcal{GP}(m(\cdot), k(\cdot, \cdot)) \tag{7.2}$$

In this work, we want to use the Gaussian Process to predict the function value for a test point x_*. Suppose we have a training set $X = \{(x_i, f(x_i)|i = 1, ..., n\}$, then the function value f_* at point x_* will have distribution:

$$f_*|x_*, X \sim \mathcal{N}(K(x_*, X)K(X, X)^{-1}f, K(x_*, x_*) - K(x_*, X)K(X, X)^{-1}K(X, x_*)) \tag{7.3}$$

where f is a vector representing all the function values, and K is the matrix composed of elements representing the covariance function $k(\cdot, \cdot)$. From this equation we can see that the GP provides not just information about the likely value of the function f, but importantly also about the uncertainty around this value.

7.2.2 Acquisition Functions for Bayesian Optimization

We assume that the function of interest $f(x)$ is drawn from a Gaussian process prior. Together with the observation data $X = \{(x_i, f(x_i)|i = 1, ..., n\}$ we can induce a posterior over functions. The acquisition function $a(x)$ determines what point in the variable space should be evaluated next via a proxy optimization $x_{next} = \arg\max_x a(x)$, which is our current best guess for the global optima. The following are several popular choices of acquisition function.

Probability of Improvement (PI) is proposed [Kushner (1964)] to maximize the probability of improving over the best current value. Alternatively, one could also choose to maximize the Expected Improvement (EI) over the current best. A more recent development is Upper Confidence Bound (UCB) [Srinivas et al. (2009)] which aims to minimize the regret over the optimization process. The UCB acquisition function has the following form:

$$a_{UCB}(x|X) = \mu(x|X) + \kappa\sigma(x|X) \tag{7.4}$$

where a tunable κ is able to balance exploration and exploitation. In this work we will focus on the UCB criterion, as in our problem we prefer to have the ability to tune the balance of exploration and exploitation.

7.2.3 Bayesian Optimization with Gaussian Process Regression

Bayesian optimization consists of two main components: a Bayesian statistical model that represents the objective, and an acquisition function for deciding where to sample next. After evaluating the objective according to an initial space-filling experimental design, often consisting of points chosen uniformly at random, they are used iteratively to allocate the remainder of a budget of N function evaluations.

Figure 7.1 shows an example of using Bayesian optimization on a 1D optimization problem [Brochu et al. (2010)]. The figures show a Gaussian process (GP) approximation of the objective function over four iterations of sampled values of the objective function. The figure also shows the acquisition function in the lower shaded plots. The acquisition is high where the GP predicts a high objective (exploitation) and where the prediction uncertainty is high (exploration)—areas with both attributes are sampled first. Note that the area on the far left remains unsampled, as while it has high uncertainty, it is (correctly) predicted to offer little improvement over the highest observation.

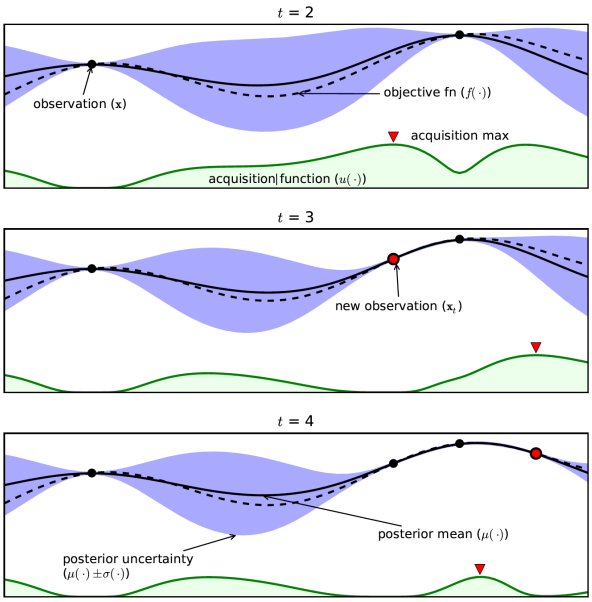


Figure 7.1: An example of using Bayesian optimization on a toy 1D optimization problem [Brochu et al. (2010)].

7.3 Problem Formulation

The aim of this work is to address the following questions:

1. Given a UTM decision-making system, can we automatically discover what scenarios lead to that system performing poorly?

To formulate this problem, consider a UTM system which consists of two parts: the scenario parameterized by the variables x_s which represents the planned flight requests, the decision-making system parameterized by the variables y_s which is responsible for generating path waypoints for the whole trajectory of the planned flight requests. Given x_s and y_s we can run simulations to evaluate a configuration of our decision-making system on a given scenario. We represent the metric we care about by m. Formally, we have:

$$m = f(\mathbf{x_s}, \mathbf{y_s}) \tag{7.5}$$

where f is the function we'd like to approximate so we can determine the value of interest m given x_s and y_s.

In this work we answer question one by performing a simulation of scenarios parameterized by x_s under decision-making system parameterized by y_s. Our goal is to find a subset(s) of x_s that lead to some range of values m given a fixed trajectory planning algorithm with parameters y_s. These will likely result in scenarios that are unsafe or inefficient, i.e. through simulation we determine they fall below some safety and/or efficiency threshold. To answer question two, our goal is to find y_s that optimizes m for a given subset of x_s. In other words, we want to find the hyper-parameters that lead to the best performing algorithm for a given set of scenarios. The above two tasks can be formulated into optimization problems and in this work we will use the Bayesian Optimization algorithm to solve the formulated optimization problem. The advantage of Bayesian Optimization is sample efficiency (the algorithm will search the most promising variable configuration given the previous information), since the function f is expensive to evaluate as we must perform a simulation to do so.

7.3.1 Failure Mode Discovery

Our first task is as follows: given a parameterized autonomous system in the UTM ecosystem, we try to identify its failure modes by maximizing the objective function that correlates with system failure. For the strategic deconfliction services considered in this work, we construct the objective

function using the sum of all NMACs in a scenario weighted by their severity:

$$f = \sum_{i=1}^{N_{\text{nmac}}} s_i \tag{7.6}$$

where s_i, the severity of NMAC i, is a function of its duration and the proximity of the other vehicle in conflict. The optimization problem then takes the following form:

$$\begin{aligned} \text{max} \quad & f(\mathbf{x_s}; \mathbf{y_s}) && \text{(7.7a)} \\ \text{subject to} \quad & g_i(\mathbf{x_s}) \leq 0, \;\; i = 1, 2, \ldots, n_{ineq} && \text{(7.7b)} \\ & h_i(\mathbf{x_s}) = 0, \;\; i = 1, 2, \ldots, n_{eq} && \text{(7.7c)} \end{aligned}$$

where g_i and h_i capture any inequality and equality constraints the scenario parameterization x_s is subject to. In practice, we apply constraints to remove infeasible scenarios from the optimization search space, and typically have to convert equality constraints into inequality constraints when performing Bayesian Optimization. Note that in the optimization above, we are optimizing only over the scenario space $\mathbf{x_s}$, and not over the parameters of the system under test $\mathbf{y_s}$. This allows us to find what kind of failures exist for a given instance of the system.

7.3.2 System Optimization

Next we consider the problem of optimizing a system for a given set of scenarios that lead to system failures. This optimization problem has the form:

$$\begin{aligned} \text{min} \quad & f(\mathbf{y_s} \mid \mathbf{x_s}) && \text{(7.8a)} \\ \text{subject to} \quad & l_i(\mathbf{y_s}) \leq 0, \;\; i = 1, 2, \ldots, n_{ineq} && \text{(7.8b)} \\ & m_i(\mathbf{y_s}) = 0, \;\; i = 1, 2, \ldots, n_{eq} && \text{(7.8c)} \end{aligned}$$

where l_i and m_i capture the infeasibility constraints associated with the parameters of the system we are optimizing. While the objective is still computed using Equation 7.6, it is now being minimized in an effort to reduce or eliminate the failure modes of the system. Note that x_s takes on the form of a random variable in the optimization above that is sampled from a distribution of scenarios D_f that led to system failures $x_s \sim D_f$.

By applying the two methodologies above in tandem, we can perform failure mode discovery and system optimization to those failure modes in a single min-max optimization. This process leads to a streamlined system and scenario level analysis of complex and difficult to reach parts of the scenario and system parameter spaces.

7.4 Urban Air Mobility Case Study

This section presents results that apply the stress testing approach outlined in this chapter to a trajectory planning UTM service for a UAM use case. We also present results that demonstrate the optimization process of the hyper-parameters of the trajectory optimizer to improve its performance (e.g., minimize the number of LOS time steps) for the found failure scenarios.

We consider a simplified UAM network that follows the generic city model presented in [Kohlman and Patterson (2018); Patterson et al. (2018)]. In this generic city model, seven vertiports are distributed in a "six around one" hexagonal pattern. As shown in Figure 7.2, vertiport 1 is located in the center of the hexagon and is located equidistant from the other six vertiports at a distance of 16km, which will cover the main congestion area of New York City. Overlays of the vertiport network are shown on a Google map image of New York in Figure 7.2, which shows typical New York traffic on a Friday at 5 pm [Google (2020)].

The decision-making system we are testing is the trajectory optimization algorithm proposed in [Egorov et al. (2019)], which is responsible for generating the flight plans for all the flight requests (the flight request is the aircraft take off vertiport location, landing vertiport location, and take off time). The trajectory optimization algorithm takes flight request(s) as input and returns a single or a set of optimized 4D trajectories that separate the vehicles according to the prescribed separation standard (150 m in these use cases). However, it should be noted this formulation is able to test more general decision making systems including various pre-departure path planning algorithms and online detect and avoid algorithms. We assume there are no in-flight disturbances to the system such as external forces or intruders that could impact vehicle performance in this preliminary phase of the work.

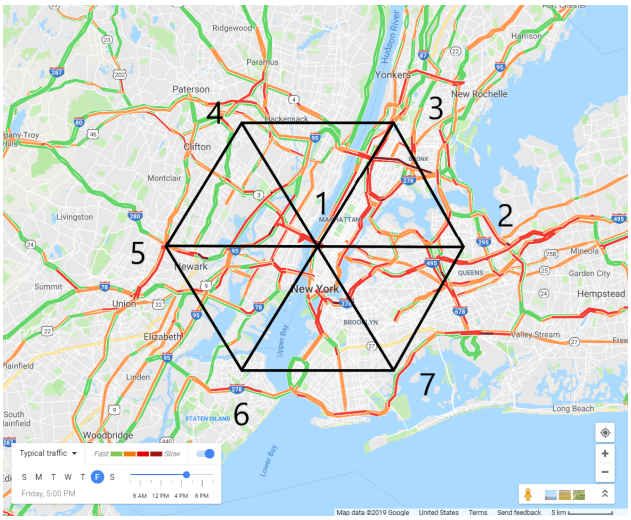
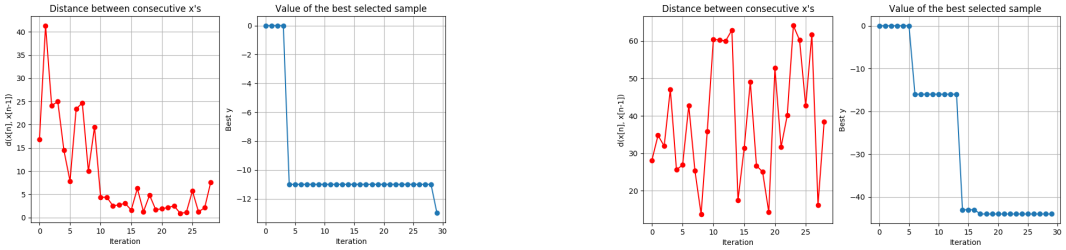


Figure 7.2: Network of seven vertiports overlaid on New York city with segment length 16km [Yang et al. (2019)].

7.4.1 Exploration and Exploitation Trade-offs

The exploration and exploitation trade-off is a critical part of the methodology, and we address it in this section. By using the Upper Confidence Bound (UCB) [Srinivas et al. (2009)] as the acquisition function in the Gaussian process optimization, we can tune the parameter responsible for this trade-off (κ in Equation 7.4). Figure 7.4 shows the effects of the exploration constant during stress testing of a trajectory optimization deconfliction service for the UAM use case operating over a hexagon network. The left plot in red color represents the Euclidean distance between consecutive sampled points. A small distance means the point we will sample next is close to the previous sampled point, which has a high mean and low variance. This usually happens when the previous point is the current optimal point. A large distance on the other hand emphasizes the exploration. As shown in Figure 7.4 (a), the algorithm gets stuck in the local optimal from sample 4 to sample 28, where the found failure scenarios have 11 LOS time steps. While in Figure 7.4 (b), the algorithm quickly finds new failure scenarios (at sample 6, 14, 17) instead of getting stuck in a local optimum. As we compare Figure 7.4 (a) and (b), it can be seen with a higher exploration weight (e.g., $\kappa = 10$), the optimization algorithm tends to focus on areas where the uncertainty is

high. This is useful for us to identify new failure modes instead of staying stuck in local optimums. Thus in the following experiment, we will set $\kappa = 10$.



(a) $\kappa = 2$ (b) $\kappa = 10$

Figure 7.4: Optimization process with different exploration weight.

7.4.2 Failure Mode Discovery and System Optimization

In this section, we present preliminary analysis of the stress-testing methodology that attempts to discover failure modes in the system under tests. We apply Gaussian process regression to the UAM use case where a single strategic deconfliction service is responsible for ensuring separation between traffic. The algorithm used for deconfliction is based on trajectory optimization, which in this configuration instance does not guarantee conflict-free trajectories. Follow on analyses will examine different variations of the deconfliction algorithm. However, in these results we consider a simpler configuration for illustrative purposes.

We perform a single pass of the algorithm on the UAM scenario described above where we sample 150 points from the scenario space of the problem. The optimization metrics during run-time are shown in Figure 7.5. Among all the 150 points, the first 50 points are sampled randomly and the other 100 points are selected sequentially by maximizing the acquisition function in Equation 7.4. As we can see in Figure 7.5, during the sampling phase for the first 75 points, the optimization algorithm is in exploration mode (the distance between consecutive sampled points is high) since the algorithm didn't find any resulting LOS event. After the optimization algorithm found the first failure case (with positive LOS time step) at around iteration 60, the algorithm gradually switched to exploitation mode, and discovers a number of other failure scenarios.

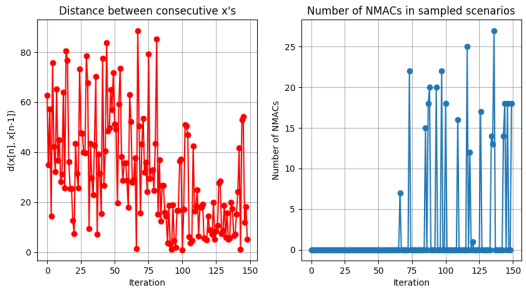


Figure 7.5: After the optimization, several failure scenarios found with positive LOS time steps by maximizing m.

We then perform a parameter optimization of the trajectory optimizer in an effort to mitigate the failures through a re-parameterization of the system. More specifically, the design variable of the path planner is of the following form:

$$y_s = [l_{\max}, l_{\min}, r, t_{\text{trials}}, t_{\text{offset}}] \tag{7.9}$$

where $l_{\max}, l_{\min}$ are the maximum/minimum length of segment in generated trajectory. r is the cost function multiplier associated with vehicle to vehicle deconfliction. $t_{\text{trials}}, t_{\text{offset}}$ are the number of additional plans generated at different times and the time offset for the additional plans. These variables are from the path algorithm and readers can refer [Egorov et al. (2019)] for more details.

Table 7.1: Failure mode types of the trajectory optimization deconfliction service, capability of the system optimization step, and potential mitigations of the failure mode if the system optimization is not able to resolve the failure.

Failure Type	Resolved Through Optimization	Failure Mitigations
During Take-Off	No	Procedures/Scheduling
During Landing	No	Procedures/Scheduling
Flying Directly Towards	Yes	-
Flying in Sequence	Yes	-
Crossing Traffic	Yes	-

Following the failure mode discovery, and system optimization steps, we classified the failure scenarios found by the stress testing algorithm into 5 categories. These categories are outlined in Table 7.1, along with indicators for whether the system optimization was successful in resolving

the failure mode. Two examples of successful system parameter optimizations that resolved the flying in sequence and crossing traffic failure modes are shown in Figure 7.6. For the two classes of failures where the system optimization was not successful, we propose an additional mitigation mechanism for the failure. Specifically, because the two failures that remained unresolved occur during take-off or landing, we propose that these two conflict types be resolved using procedural separation or scheduling services.

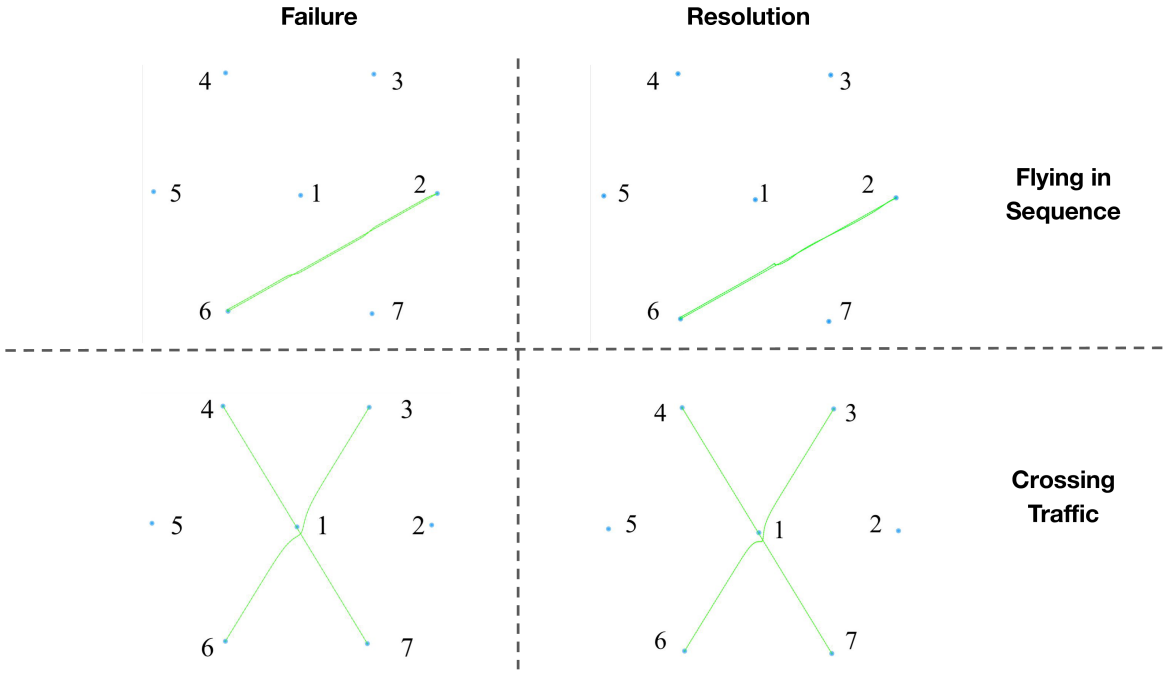


Figure 7.6: Examples of system parameter optimization that successfully resolved two safety-critical failure modes.

7.5 Conclusion

In this work, we developed a sample efficient algorithm which can sequentially sample points to optimize the objective function. Given a specific UTM use case, this algorithm can be used for stress testing by maximizing the number of LOS time steps or for optimizing the path planner to improve its performance by minimizing the number of LOS time steps. A hexagon vertiport is

built in the prototype simulator for testing this algorithm. The developed algorithm successfully identified five different types of failure scenarios given the airspace configuration, as well as improved the performance of the path planner in the five failure scenarios above. Besides, the developed algorithm is flexible to incorporate any desired constraints/use cases/metrics. And it's able to test other more general decision-making systems (pre-departure path planning algorithm or onboard detect-and-avoid algorithm) as well.

CHAPTER 8. CONCLUSIONS AND FUTURE WORK

The increase in road traffic congestion in urban areas leads to an interest in Urban Air Mobility (UAM), where the electric vertical take-off and landing (eVTOL) aircraft is able to alleviate transportation congestion by utilizing 3D airspace efficiently. To accommodate the large-scale complex systems in UAM and enable safe and scalable flight operations, a shift from human-centric air traffic control systems towards higher levels of autonomy is required. However, current research work faces scalability challenges to be applied in UAM, where is a dynamic complex environment with multiple agents interacting with each other.

In this dissertation, we propose a safe and verified learning-based decentralized perception and decision-making system for eVTOLs to operate in high-density air traffic conditions, which could enable safe, efficient, and scalable flight operations in UAM.

8.1 Contributions

The contributions of this dissertation are now summarized:

1. For the learning-based perception algorithm, we propose a deep-learning based computer vision model for detection and tracking. In contrast with existing research with ground-level pedestrian detection, the developed algorithm achieves highly accurate multiple pedestrian detection from a bird-eye view, when both the pedestrians and the aircraft platform are moving.

2. For the aircraft guidance and control, a decentralized message-passing based computational guidance algorithm with separation assurance is proposed using Multiagent Markov Decision Process (MMDP) and Monte Carlo Tree Search (MCTS) algorithm. We use a logit level-k model for the multi-aircraft coordination based on the message-passing mechanism through

wireless communication. To achieve higher scalability, we introduce the airspace sector concept into the UAM environment by dividing the airspace into sectors, so that each aircraft only needs to coordinate with aircraft in the same sector. Besides, we also consider the communication constraints among the aircraft to make the proposed algorithm more practical, by modifying the proposed computational guidance algorithms given certain communication constraints (time, bandwidth, and communication loss) and designing air-to-air and air-to-ground communication frameworks to facilitate the computational guidance algorithm.

3. Finally, to validate the proposed autonomous decision-making systems, we propose a novel and sample efficient approach for adaptive stress testing that uses Bayesian optimization, expand on the approach in a way that enables the methodology to also optimize the system with respect to the discovered failure modes, and apply this approach to validate and optimize strategic decision-making services for the UAM and the package delivery uses cases.

8.2 Future Work

In this dissertation, the proposed concept of operations and algorithm provide a potential solution for decentralized separation assurance to enable safe, efficient, and scalable flight operations in on-demand urban air transportation with high-density air traffic. We found this framework might be a potential solution for certain airspace such as rural or suburban areas. However, the proposed framework is still in the exploratory phase with some limitations. To make the proposed algorithm more practical in real-world applications, future work should include the following directions:

1. In the current work, we assume full observability for the state information, where the aircraft can sense the other aircraft information (position and velocity) perfectly. However in practice when the state information becomes imperfect, some techniques such as Kalman Filter or Partially Observable MDP will be necessary to filter out the sensor noise.

2. Currently the computational guidance algorithm can only adopt several discrete actions at the horizontal level. Incorporating a high-fidelity aircraft dynamics model with a more realistic

mission profile and expanding the action space include more actions and altitude changes would make the proposed algorithm more practical.

3. When conduction numerical experiments in the simulator, we do not consider any restricted airspace or stationary obstacles. Adapting the proposed algorithm in restricted airspace or structured airspace in the urban area would be an interesting future direction.

4. Note that in the numerical experiments, the recorded number of NMACs during simulation is in the absence of a collision avoidance system such as TCAS or ACAS-X. A direction of future work would be to integrate our separation assurance model with a collision avoidance system as the final layer of protection, or any other strategic flight plan deconfliction, traffic flow management, and flow control techniques.

5. For the aircraft perception algorithm, currently the detection time for the trained model is around 70ms per image. To achieve real-time pedestrian detection, future work includes using model pruning to reduce the model size and increase inference speed, since previous research shows that 90% of network connections can be removed without degrading network performance. With the pruned model, it's possible to deploy the trained model on a mobile GPU Nvidia Maxwell.

6. For the proposed adaptive stress testing algorithm, future work beyond this dissertation includes trying to incorporate multiple aircraft in the scenarios (more than 2) and try to identify more failure scenarios between multiple aircraft. Also noise will be incorporated in this developed algorithm to deal with the noisy observation. Finally, other metrics (efficiency, delay, power consumption, etc.) are possible to incorporate into the proposed framework by formulating this problem as a multi-objective optimization problem.

BIBLIOGRAPHY

Airbus (2016). Acubed by airbus group. https://www.airbus-sv.com/projects/1. Accessed: 2018-11-25.

Airbus (2018). Vahana. https://www.airbus.com/innovation/urban-air-mobility/vehicle-demonstrators/vahana.html. Accessed: 2020-02-15.

Airbus UTM (2017). Future of urban mobility. https://www.airbus.com/newsroom/news/en/2016/12/My-Kind-Of-Flyover.html. Accessed: 2020-02-15.

Alejo, D., Cobano, J., Heredia, G., and Ollero, A. (2014). Optimal reciprocal collision avoidance with mobile and static obstacles for multi-uav systems. In *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pages 1259–1266. IEEE.

Allignol, C., Barnier, N., Durand, N., Manfredi, G., and Blond, É. (2017). Assessing the robustness of a uas detect & avoid algorithm. In *12th USA/Europe Air Traffic Management Research and Development Seminar*.

Ashok, P., Brázdil, T., Křetínskỳ, J., and Slámečka, O. (2018). Monte carlo tree search for verifying reachability in markov decision processes. In *International Symposium on Leveraging Applications of Formal Methods*, pages 322–335. Springer.

Augugliaro, F., Schoellig, A. P., and D'Andrea, R. (2012). Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1917–1922. IEEE.

Barhydt, R., Kopardekar, P., Battiste, V., Doble, N., Johnson, W., Lee, P., Prevot, T., and Smith, N. (2005). Joint nasa ames/langley experimental evaluation of integrated air/ground operations for en route free maneuvering.

Battiste, V., Johnson, W., Kopardekar, P., Lozito, S., Mogford, R., and Palmer, E. (2002). Distributed air/ground traffic management-technology and concept demonstration report. In *AIAA's Aircraft Technology, Integration, and Operations (ATIO) 2002 Technical Forum*, page 5825.

Bellman, R. (1957). A markovian decision process. *Indiana Univ. Math. J.*, 6:679–684.

Bertram, J. and Wei, P. (2020a). Distributed computational guidance for high-density urban air mobility with cooperative and non-cooperative collision avoidance. In *AIAA Scitech 2020 Forum*, page 1371.

Bertram, J. and Wei, P. (2020b). An efficient algorithm for self-organized terminal arrival in urban air mobility. In *AIAA Scitech 2020 Forum*, page 0660.

Bilimoria, K. D., Grabbe, S. R., Sheth, K. S., and Lee, H. Q. (2003). Performance evaluation of airborne separation assurance for free flight. *Air Traffic Control Quarterly*, 11(2):85–102.

Blom, H. A. and Bakker, G. (2015). Safety evaluation of advanced self-separation under very high en route traffic demand. *Journal of Aerospace Information Systems*, 12(6):413–427.

Bosson, C. and Lauderdale, T. A. (2018). Simulation evaluations of an autonomous urban air mobility network management and separation service. In *2018 Aviation Technology, Integration, and Operations Conference*, page 3365.

Boutilier, C. (1996). Planning, learning and coordination in multiagent decision processes. In *Proceedings of the 6th conference on Theoretical aspects of rationality and knowledge*, pages 195–210. Morgan Kaufmann Publishers Inc.

Boutilier, C. (1999). Sequential optimality and coordination in multiagent systems. In *IJCAI*, volume 99, pages 478–485.

Boutilier, C., Dean, T., and Hanks, S. (1999). Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94.

Brittain, M. and Wei, P. (2019). Autonomous separation assurance in an high-density en route sector: A deep multi-agent reinforcement learning approach. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3256–3262. IEEE.

Brochu, E., Cora, V. M., and De Freitas, N. (2010). A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*.

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. (2012). A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43.

Bulusu, V., Sengupta, R., Mueller, E. R., and Xue, M. (2018). A throughput based capacity metric for low-altitude airspace. In *2018 Aviation Technology, Integration, and Operations Conference*, page 3032.

Busoniu, L., Babuška, R., and De Schutter, B. (2010). Multi-agent reinforcement learning: An overview. *Innovations in multi-agent systems and applications-1*, 310:183–221.

Camerer, C. F. (2011). *Behavioral game theory: Experiments in strategic interaction*. Princeton University Press.

Canny, J. (1988). *The complexity of robot motion planning*. MIT press.

Čáp, M., Novák, P., Vokrínek, J., and Pěchouček, M. (2013). Multi-agent rrt: sampling-based cooperative pathfinding. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1263–1264. International Foundation for Autonomous Agents and Multiagent Systems.

Champandard, A. J. (2014). Monte-carlo tree search in total war: Rome ii's campaign ai. *AIGameDev.com: http://aigamedev.com/open/coverage/mcts-rome-ii*.

Chaslot, G. M. J., Winands, M. H., HERIK, H. J. V. D., Uiterwijk, J. W., and Bouzy, B. (2008). Progressive strategies for monte-carlo tree search. *New Mathematics and Natural Computation*, 4(03):343–357.

Chen, Y. F., Liu, M., Everett, M., and How, J. P. (2017). Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 285–292. IEEE.

Clari, M. S. V., Ruigrok, R. C., Hoekstra, J. M., and Visser, H. G. (2001). Cost-benefit study of free flight with airborne separation assurance. *Air Traffic Control Quarterly*, 9(4):287–309.

Cobano, J. A., Conde, R., Alejo, D., and Ollero, A. (2011). Path planning based on genetic algorithms and the monte-carlo method to avoid aerial vehicle collisions under uncertainties. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4429–4434. IEEE.

Connolly, C. I., Burns, J. B., and Weiss, R. (1990). Path planning using laplace's equation. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 2102–2106. IEEE.

Conroy, P., Bareiss, D., Beall, M., and Berg, J. v. d. (2014). 3-d reciprocal collision avoidance on physical quadrotor helicopters with on-board sensing for relative positioning. *arXiv preprint arXiv:1411.3794*.

Consiglio, M., Hoadley, S., Wing, D., and Baxley, B. (2007). Safety performance of airborne separation: Preliminary baseline testing. In *7th AIAA ATIO Conf, 2nd CEIAT Int'l Conf on Innov and Integr in Aero Sciences, 17th LTA Systems Tech Conf; followed by 2nd TEOS Forum*, page 7739.

Consiglio, M., Muñoz, C., Hagen, G., Narkawicz, A., and Balachandran, S. (2016). Icarous: Integrated configurable algorithms for reliable operations of unmanned systems. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pages 1–5. IEEE.

Cook, S. P. and Brooks, D. (2015). A quantitative metric to enable unmanned aircraft systems to remain well clear. *Air Traffic Control Quarterly*, 23(2-3):137–156.

Couëtoux, A., Hoock, J.-B., Sokolovska, N., Teytaud, O., and Bonnard, N. (2011). Continuous upper confidence trees. In *International Conference on Learning and Intelligent Optimization*, pages 433–445. Springer.

Coulom, R. (2006). Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer.

Coulom, R. (2007). Computing "elo ratings" of move patterns in the game of go. *Icga Journal*, 30(4):198–208.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection.

Daskalakis, C., Goldberg, P. W., and Papadimitriou, C. H. (2009). The complexity of computing a nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259.

David J., W., Richard J., A., Jacqueline A., D., Brain M., L., Bryan E., B., and Donald, M. (2001). Airborne use of traffic intent information in a distributed air-ground traffic management concept: Experiment design and preliminary results.

Delahaye, D., Peyronne, C., Mongeau, M., and Puechmorel, S. (2010). Aircraft conflict resolution by genetic algorithm and b-spline approximation. In *EIWAC 2010, 2nd ENRI International Workshop on ATM/CNS*, pages 71–78.

Desaraju, V. R. and How, J. P. (2011). Decentralized path planning for multi-agent teams in complex environments using rapidly-exploring random trees. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4956–4961. IEEE.

Egorov, M., Kuroda, V., and Sachs, P. (2019). Encounter aware flight planning in the unmanned airspace. In *2019 Integrated Communications, Navigation and Surveillance Conference (ICNS)*, pages 1–15. IEEE.

Enright, P. J. and Conway, B. A. (1992). Discrete approximations to optimal trajectories using direct transcription and nmiscar programming. *Journal of Guidance, Control, and Dynamics*, 15(4):994–1002.

Even-Dar, E., Kearns, M., and Wortman, J. (2006). Risk-sensitive online learning. In *International Conference on Algorithmic Learning Theory*, pages 199–213. Springer.

Federal Aviation Administration (2015). Unmanned aerial system traffic management fact sheet. https://www.nasa.gov/ames/utm2015. Accessed: 2018-01-19.

Federal Aviation Administration (2016). Nextgen implementation plan 2016. U.S. Department of Transportation.

Federal Aviation Administration (2017). Near midair collision reporting. http://www.faraim.org/aim/aim-4-03-14-530.html. Accessed: 2020-02-15.

Federal Aviation Administration (2019). Control sector. http://www.faraim.org/aim/aim-4-03-14-618.html. Accessed: 2020-02-15.

Feinberg, E. A. and Shwartz, A. (2012). *Handbook of Markov decision processes: methods and applications*, volume 40. Springer Science & Business Media.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2009). Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645.

Fiorini, P. and Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–772.

Force, R. T. (1995). Final report of rtca task force 3: Free flight implementation. Technical report, RTCA Inc., Tech. rep.

Frazzoli, E., Mao, Z.-H., Oh, J.-H., and Feron, E. (2001). Resolution of conflicts involving many aircraft via semidefinite programming. *Journal of Guidance, Control, and Dynamics*, 24(1):79–86.

Frew, E., McGee, T., Kim, Z., Xiao, X., Jackson, S., Morimoto, M., Rathinam, S., Padial, J., and Sengupta, R. (2004). Vision-based road-following using a small autonomous aircraft. In *2004 IEEE Aerospace Conference Proceedings (IEEE Cat. No. 04TH8720)*, volume 5, pages 3006–3015. IEEE.

Gawdiak, Y., Carr, G., and Hasan, S. (2009). Jpdo case study of nextgen high density operations. In *9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO) and Aircraft Noise and Emissions Reduction Symposium (ANERS)*, page 6918.

Gawdiak, Y., Holmes, B., Sawhill, B., Herriot, J., Ballard, D., Creedon, J., Eckhause, J., Long, D., Hemm, R., Murphy, C., et al. (2012). Air transportation strategic trade space modeling and assessment through analysis of on-demand air mobility with electric aircraft. In *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, page 5594.

Gipson, L. (2017). Nasa embraces urban air mobility, calls for market study. https://www.nasa.gov/aero/nasa-embraces-urban-air-mobility. Accessed: 2020-02-15.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

Google (2020). Google maps. https://www.google.com/maps. Accessed: 2020-02-15.

Han, S.-C., Bang, H., and Yoo, C.-S. (2009). Proportional navigation-based collision avoidance for uavs. *International Journal of Control, Automation and Systems*, 7(4):553–565.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hoekstra, J. M., van Gent, R. N., and Ruigrok, R. C. (2002). Designing for safety: the 'free flight' air traffic management concept. *Reliability Engineering & System Safety*, 75(2):215–232.

Hoffmann, G., Rajnarayan, D. G., Waslander, S. L., Dostal, D., Jang, J. S., and Tomlin, C. J. (2004). The stanford testbed of autonomous rotorcraft for multi agent control (starmac). In *The 23rd Digital Avionics Systems Conference (IEEE Cat. No. 04CH37576)*, volume 2, pages 12–E. IEEE.

Holden, J. and Goel, N. (2016). Fast-forwarding to a future of on-demand urban air transportation. *San Francisco, CA*.

Howard, R. A. (1964). Dynamic programming and markov processes.

Howlet, J. K., Schulein, G., and Mansur, M. H. (2004). A practical approach to obstacle field route planning for unmanned rotorcraft.

Huang, X., Kwiatkowska, M., Wang, S., and Wu, M. (2017). Safety verification of deep neural networks. In *International Conference on Computer Aided Verification*, pages 3–29. Springer.

Inalhan, G., Stipanovic, D. M., and Tomlin, C. J. (2002). Decentralized optimization, with application to multiple aircraft coordination. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 1, pages 1147–1155. IEEE.

Ingersoll, K., Niedfeldt, P. C., and Beard, R. W. (2015). Multiple target tracking and stationary object detection in video with recursive-ransac and tracker-sensor feedback. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1320–1329. IEEE.

Jeannin, J.-B., Ghorbal, K., Kouskoulas, Y., Gardner, R., Schmidt, A., Zawadzki, E., and Platzer, A. (2015). Formal verification of acas x, an industrial airborne collision avoidance system. In *2015 International Conference on Embedded Software (EMSOFT)*, pages 127–136. IEEE.

Jones, D. R. (2001). A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, 21(4):345–383.

Julian, K. D. and Kochenderfer, M. J. (2017). Neural network guidance for uavs. In *AIAA Guidance, Navigation, and Control Conference*, page 1743.

Julian, K. D., Kochenderfer, M. J., and Owen, M. P. (2018). Deep neural network compression for aircraft collision avoidance systems. *Journal of Guidance, Control, and Dynamics*, 42(3):598–608.

Julian, K. D., Lopez, J., Brush, J. S., Owen, M. P., and Kochenderfer, M. J. (2016). Policy compression for aircraft collision avoidance systems. In *Digital Avionics Systems Conference (DASC), 2016 IEEE/AIAA 35th*, pages 1–10. IEEE.

Jung, J., D'Souza, S. N., Johnson, M. A., Ishihara, A. K., Modi, H. C., Nikaido, B., and Hasseeb, H. (2016). Applying required navigation performance concept for traffic management of small unmanned aircraft systems.

Kahn, G., Zhang, T., Levine, S., and Abbeel, P. (2017). Plato: Policy learning using adaptive trajectory optimization. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3342–3349. IEEE.

Kahne, S. and Frolow, I. (1996). Air traffic management: Evolution with technology. *IEEE Control Systems*, 16(4):12–21.

Kanellakis, C. and Nikolakopoulos, G. (2017). Survey on computer vision for uavs: Current developments and trends. *Journal of Intelligent & Robotic Systems*, 87(1):141–168.

Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894.

Karr, D., Vivona, R., Roscoe, D., Depascale, S., and Wing, D. (2012). Autonomous operations planner: A flexible platform for research in flight-deck support for airborne self-separation. In *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, page 5417.

Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. (2017). Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer.

Kavraki, L., Svestka, P., and Overmars, M. H. (1994). *Probabilistic roadmaps for path planning in high-dimensional configuration spaces*, volume 1994. Unknown Publisher.

Khatib, O. and Mampey, L. (1978). Fonction decision-commande d'un robot manipulateur. *Rep*, 2(7):156.

Kleinbekman, I. C., Mitici, M. A., and Wei, P. (2018). evtol arrival sequencing and scheduling for on-demand urban air mobility. In *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, pages 1–7. IEEE.

Kochenderfer, M. J. (2015). *Decision making under uncertainty: theory and application*. MIT press.

Kochenderfer, M. J. and Chryssanthacopoulos, J. (2011). Robust airborne collision avoidance through dynamic programming. *Massachusetts Institute of Technology, Lincoln Laboratory, Project Report ATC-371*.

Kochenderfer, M. J., Holland, J. E., and Chryssanthacopoulos, J. P. (2012). Next-generation airborne collision avoidance system. Technical report, Massachusetts Institute of Technology-Lincoln Laboratory Lexington United States.

Kocsis, L. and Szepesvári, C. (2006). Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer.

Kocsis, L., Szepesvári, C., and Willemson, J. (2006). Improved monte-carlo search. *Univ. Tartu, Estonia, Tech. Rep*, 1.

Koditschek, D. E. and Rimon, E. (1990). Robot navigation functions on manifolds with boundary. *Advances in applied mathematics*, 11(4):412–442.

Koenig, S. and Simmons, R. (1998). Xavier: A robot navigation architecture based on partially observable markov decision process models. *Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems*, pages 91–122.

Kohlman, L. W. and Patterson, M. D. (2018). System-level urban air mobility transportation modeling and determination of energy-related constraints. In *2018 Aviation Technology, Integration, and Operations Conference*, page 3677.

Kopardekar, P., Rios, J., Prevot, T., Johnson, M., Jung, J., and Robinson, J. E. (2016). Unmanned aircraft system traffic management (utm) concept of operations.

Koren, M., Alsaif, S., Lee, R., and Kochenderfer, M. J. (2018). Adaptive stress testing for autonomous vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–7. IEEE.

Krozel, J., Peters, M., and Bilimoria, K. (2000). A decentralized control strategy for distributed air/ground traffic separation. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 4062.

Kuchar, J. K. and Yang, L. C. (2000). A review of conflict detection and resolution modeling methods. *IEEE Transactions on intelligent transportation systems*, 1(4):179–189.

Kushner, H. J. (1964). A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106.

LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning.

Lee, R., Kochenderfer, M. J., Mengshoel, O. J., Brat, G. P., and Owen, M. P. (2015). Adaptive stress testing of airborne collision avoidance systems. In *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, pages 6C2–1. IEEE.

Lee, R., Mengshoel, O. J., and Kochenderfer, M. J. (2019). Adaptive stress testing of safety-critical systems. In *Safe, Autonomous and Intelligent Vehicles*, pages 77–95. Springer.

Li, H., Dai, H., and Li, C. (2008). Communication complexity for distributed scheduling in wireless ad hoc networks. *submitted to Allerton*.

Li, S., Egorov, M., and Kochenderfer, M. (2019). Optimizing collision avoidance in dense airspace using deep reinforcement learning. *arXiv preprint arXiv:1912.10146*.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017a). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017b). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Liu, C., Arnon, T., Lazarus, C., Barrett, C., and Kochenderfer, M. J. (2019). Algorithms for verifying deep neural networks. *arXiv preprint arXiv:1903.06758*.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer.

Lötjens, B., Everett, M., and How, J. P. (2019). Safe reinforcement learning with model uncertainty estimates. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8662–8668. IEEE.

Lovering, Z. (2016). Vahana configuration trade study - part i – vahana. https://vahana.aero/vahana-configuration-trade-study-part-i-47729eed1cdf.

Mariton, M. (1990). *Jump linear systems in automatic control*. M. Dekker New York.

Mellinger, D., Kushleyev, A., and Kumar, V. (2012). Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 477–483. IEEE.

Mihatsch, O. and Neuneier, R. (2002). Risk-sensitive reinforcement learning. *Machine learning*, 49(2-3):267–290.

Mitici, M. and Blom, H. A. (2018). Mathematical models for air traffic conflict and collision probability estimation. *IEEE Transactions on Intelligent Transportation Systems*, 20(3):1052–1068.

Mockus, J. (2012). *Bayesian approach to global optimization: theory and applications*, volume 37. Springer Science & Business Media.

Molloy, T. L., Ford, J. J., and Mejias, L. (2017). Detection of aircraft below the horizon for vision-based detect and avoid in unmanned aircraft systems. *Journal of Field Robotics*, 34(7):1378–1391.

Moore, M. (2017). Uber elevate: evtol urban mobility. *Rotorcraft Business & Technology Summit*.

Moore, M. D. and Goodrich, K. H. (2013). High speed mobility through on-demand aviation. In *2013 Aviation Technology, Integration, and Operations Conference*, page 4373.

Morgan, D., Chung, S.-J., and Hadaegh, F. Y. (2014). Model predictive control of swarms of spacecraft using sequential convex programming. *Journal of Guidance, Control, and Dynamics*, 37(6):1725–1740.

Mueller, E. R., Kopardekar, P. H., and Goodrich, K. H. (2017). Enabling airspace integration for high-density on-demand mobility operations. In *17th AIAA Aviation Technology, Integration, and Operations Conference*, page 3086.

Muñoz, C., Narkawicz, A., Hagen, G., Upchurch, J., Dutle, A., Consiglio, M., and Chamberlain, J. (2015). Daidalus: detect and avoid alerting logic for unmanned systems. pages 5A1–1–5A1–12.

Nash, J. F. et al. (1950). Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, 36(1):48–49.

NATS (2019). Introduction to airspace. https://www.nats.aero/ae-home/introduction-to-airspace/. Accessed: 2020-02-15.

Netjasov, F. and Janic, M. (2008). A review of research on risk and safety modelling in civil aviation. *Journal of Air Transport Management*, 14(4):213–220.

Niedfeldt, P. C. and Beard, R. W. (2014). Multiple target tracking using recursive ransac. In *2014 American Control Conference*, pages 3393–3398. IEEE.

Niedfeldt, P. C. and Beard, R. W. (2015). Convergence and complexity analysis of recursive-ransac: A new multiple target tracking algorithm. *IEEE Transactions on Automatic Control*, 61(2):456–461.

Niedfeldt, P. C., Ingersoll, K., and Beard, R. W. (2017). Comparison and analysis of recursive-ransac for multiple target tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 53(1):461–476.

Ong, H. Y. and Kochenderfer, M. J. (2016). Markov decision process-based distributed conflict resolution for drone air traffic management. *Journal of Guidance, Control, and Dynamics*, pages 69–80.

Orjih, O. (2006). Recent developments in aircraft wireless networks. *dated Apr*, 23.

Ozge Unel, F., Ozkalayci, B. O., and Cigla, C. (2019). The power of tiling for small object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0.

Pallottino, L., Feron, E. M., and Bicchi, A. (2002). Conflict resolution problems for air traffic management systems solved with mixed integer programming. *IEEE transactions on intelligent transportation systems*, 3(1):3–11.

Pallottino, L., Scordio, V. G., Frazzoli, E., and Bicchi, A. (2006). Probabilistic verification of a decentralized policy for conflict resolution in multi-agent systems. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2448–2453. IEEE.

Park, J.-W., Oh, H.-D., and Tahk, M.-J. (2008). Uav collision avoidance based on geometric approach. In *SICE Annual Conference, 2008*, pages 2122–2126. IEEE.

Patterson, M. D., Antcliff, K. R., and Kohlman, L. W. (2018). A proposed approach to studying urban air mobility missions including an initial exploration of mission requirements.

Pontani, M. and Conway, B. A. (2010). Particle swarm optimization applied to space trajectories. *Journal of Guidance, Control, and Dynamics*, 33(5):1429–1441.

Porsche Consulting study (2018). the future of vertical mobility. https://www.porsche-consulting.com/fileadmin/docs/04_Medien/Publikationen/TT1371_The_Future_of_Vertical_Mobility/The_Future_of_Vertical_Mobility_A_Porsche_Consulting_study__C_2018.pdf. Accessed: 2020-02-15.

Pradeep, P. and Wei, P. (2018a). Energy optimal speed profile for arrival of tandem tilt-wing evtol aircraft with rta constraint. *IEEE CSAA Guidance, Navigation and Control Conference*.

Pradeep, P. and Wei, P. (2018b). Heuristic approach for arrival sequencing and scheduling for evtol aircraft in on-demand urban air mobility. In *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, pages 1–7. IEEE.

Pradeep, P. and Wei, P. (2019). Energy-efficient arrival with rta constraint for multirotor evtol in urban air mobility. *Journal of Aerospace Information Systems*, 16(7):263–277.

Purwin, O., D'Andrea, R., and Lee, J.-W. (2008). Theory and implementation of path planning by negotiation for decentralized agents. *Robotics and Autonomous Systems*, 56(5):422–436.

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Raghunathan, A. U., Gopal, V., Subramanian, D., Biegler, L. T., and Samad, T. (2004). Dynamic optimization strategies for three-dimensional conflict resolution of multiple aircraft. *Journal of guidance, control, and dynamics*, 27(4):586–594.

Raju, D., Bharadwaj, S., and Topcu, U. (2019). Decentralized runtime synthesis of shields for multi-agent systems. *arXiv preprint arXiv:1910.10380*.

Rappaport, T. S. et al. (1996). *Wireless communications: principles and practice*, volume 2. prentice hall PTR New Jersey.

Rasmussen, C. E. (2003). Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.

Richards, A. and How, J. (2004). Decentralized model predictive control of cooperating uavs. In *43rd IEEE Conference on Decision and Control*, volume 4, pages 4286–4291. Citeseer.

Richards, A. and How, J. P. (2002). Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *American Control Conference, 2002. Proceedings of the 2002*, volume 3, pages 1936–1941. IEEE.

Rimon, E. and Koditschek, D. E. (1988). Exact robot navigation using cost functions: the case of distinct spherical boundaries in e/sup n. In *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pages 1791–1796. IEEE.

Ringwald, T., Sommer, L., Schumann, A., Beyerer, J., and Stiefelhagen, R. (2019). Uav-net: A fast aerial vehicle detector for mobile platforms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0.

Sakamaki, J. Y., Beard, R. W., and Rice, M. (2017). Tracking multiple ground objects using a team of unmanned air vehicles. In *Sensing and Control for Autonomous Vehicles*, pages 249–268. Springer.

Saripalli, S. (2009). Vision-based autonomous landing of an helicopter on a moving target. In *AIAA guidance, navigation, and control conference*, page 5660.

SC-186, R. F. (2006). *Minimum Operational Performance Standards for 1090 MHz Extended Squitter: Automatic Dependent Surveillance-Broadcast (ADS-B) and Traffic Information Services-Broadcast (TIS-B)*. RTCA.

Schouwenaars, T., De Moor, B., Feron, E., and How, J. (2001). Mixed integer programming for multi-vehicle path planning. In *Control Conference (ECC), 2001 European*, pages 2603–2608. IEEE.

Schouwenaars, T., How, J., and Feron, E. (2004). Decentralized cooperative trajectory planning of multiple aircraft with hard safety guarantees. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 5141.

Shim, D. H., Kim, H. J., and Sastry, S. (2003). Decentralized nonlinear model predictive control of multiple flying robots. In *Decision and control, 2003. Proceedings. 42nd IEEE conference on*, volume 4, pages 3621–3626. IEEE.

Shim, D. H. and Sastry, S. (2007). An evasive maneuvering algorithm for uavs in see-and-avoid situations. In *American Control Conference, 2007. ACC'07*, pages 3886–3891. IEEE.

Shrivastava, A., Gupta, A., and Girshick, R. (2016). Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 761–769.

Siegwart, R., Nourbakhsh, I. R., and Scaramuzza, D. (2011). *Introduction to autonomous mobile robots*. MIT press.

Sigaud, O. and Buffet, O. (2013). *Markov decision processes in artificial intelligence*. John Wiley & Sons.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2017a). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017b). Mastering the game of go without human knowledge. *Nature*, 550(7676):354.

SKYbrary (2019a). Air-ground voice communications. https://www.skybrary.aero/index.php/Air-Ground_Voice_Communications. Accessed: 2019-11-30.

SKYbrary (2019b). Loss of separation at sector boundaries. https://www.skybrary.aero/index.php/Loss_of_Separation_at_Sector_Boundaries#. Accessed: 2020-02-15.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959.

Srinivas, N., Krause, A., Kakade, S. M., and Seeger, M. (2009). Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*.

Stahl, D. O. and Wilson, P. W. (1995). On players' models of other players: Theory and experimental evidence. *Games and Economic Behavior*, 10(1):218–254.

Stahl II, D. O. and Wilson, P. W. (1994). Experimental evidence on players' models of other players. *Journal of economic behavior & organization*, 25(3):309–327.

Stoschek, A. (2017). Exploring sense-and-avoid systems for autonomous vehicles. https://acubed.airbus.com/blog/vahana/exploring-sense-and-avoid-systems-for-autonomous-vehicles/. Accessed: 2020-02-15.

Syed, N., Rye, M., Ade, M., Trani, A., Hinze, N., Swingle, H., Smith, J. C., Dollyhigh, S., and Marien, T. (2017). Preliminary considerations for odm air traffic management based on analysis of commuter passenger demand and travel patterns for the silicon valley region of california. In *17th AIAA Aviation Technology, Integration, and Operations Conference*, page 3082.

Temizer, S., Kochenderfer, M., Kaelbling, L., Lozano-Pérez, T., and Kuchar, J. (2010). Collision avoidance for unmanned aircraft using markov decision processes. In *AIAA guidance, navigation, and control conference*, page 8040.

Tesauro, G. and Galperin, G. R. (1997). On-line policy improvement using monte-carlo search. In *Advances in Neural Information Processing Systems*, pages 1068–1074.

Thipphavong, D. P., Apaza, R., Barmore, B., Battiste, V., Burian, B., Dao, Q., Feary, M., Go, S., Goodrich, K. H., Homola, J., et al. (2018). Urban air mobility airspace integration concepts

and considerations. In *2018 Aviation Technology, Integration, and Operations Conference*, page 3676.

Thrun, S. (2002). Probabilistic robotics. *Communications of the ACM*, 45(3):52–57.

Timar, S., Hunter, G., and Post, J. (2013). Assessing the benefits of nextgen performance based navigation (pbn). In *10th USA/Europe Air Traffic Management Research and Development Seminar, Chicago, Illinois*.

Tomlin, C., Pappas, G. J., and Sastry, S. (1998). Conflict resolution for air traffic management: A study in multiagent hybrid systems. *IEEE Transactions on automatic control*, 43(4):509–521.

Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, 104(2):154–171.

Van Den Berg, J., Guy, S. J., Lin, M., and Manocha, D. (2011). Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer.

Van Wesel, P. and Goodloe, A. E. (2017). Challenges in the verification of reinforcement learning algorithms.

Vascik, P. D., Cho, J., Bulusu, V., and Polishchuk, V. (2019). A geometric approach towards airspace assessment for emerging operations.

Vela, A., Solak, S., Singhose, W., and Clarke, J.-P. (2009). A mixed integer program for flight-level assignment and speed control for conflict resolution. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 5219–5226. IEEE.

Vertical Flight Society (2018). Acubed vahana. http://evtol.news/aircraft/a3-by-airbus/.

Viola, P., Jones, M., et al. (2001). Rapid object detection using a boosted cascade of simple features. *CVPR (1)*, 1(511-518):3.

Wang, Y., Audibert, J.-Y., and Munos, R. (2009). Algorithms for infinitely many-armed bandits. In *Advances in Neural Information Processing Systems*, pages 1729–1736.

White, D. J. (1993). A survey of applications of markov decision processes. *Journal of the operational research society*, 44(11):1073–1096.

Wolf, T. B. and Kochenderfer, M. J. (2011). Aircraft collision avoidance using monte carlo real-time belief space search. *Journal of Intelligent & Robotic Systems*, 64(2):277–298.

Wollkind, S., Valasek, J., and Ioerger, T. (2004). Automated conflict resolution for air traffic management using cooperative multiagent negotiation. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 4992.

Xue, M. and Rios, J. (2017). Initial study of an effective fast-time simulation platform for unmanned aircraft system traffic management. In *17th AIAA Aviation Technology, Integration, and Operations Conference*, page 3073.

Xue, M., Rios, J., Silva, J., Zhu, Z., and Ishihara, A. K. (2018). Fe3: An evaluation tool for low-altitude air traffic operations. In *2018 Aviation Technology, Integration, and Operations Conference*, page 3848.

Yang, X., Deng, L., Liu, J., Wei, P., and Li, H. (2020). Multi-agent autonomous operations in urban air mobility with communication constraints. In *AIAA Scitech 2020 Forum*, page 1839.

Yang, X., Deng, L., and Wei, P. (2019). Multi-agent autonomous on-demand free flight operations in urban air mobility. In *AIAA Aviation 2019 Forum*, page 3520.

Yang, X. and Wei, P. (2018). Autonomous on-demand free flight operations in urban air mobility using monte carlo tree search. In *8th International Conference on Research in Air Transportation (ICRAT)*. ICRAT.

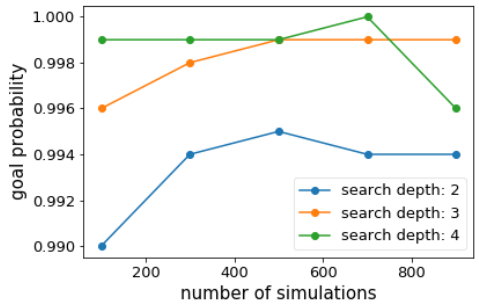
Zhang, P., Zhong, Y., and Li, X. (2019). Slimyolov3: Narrower, faster and better for real-time uav applications. *arXiv preprint arXiv:1907.11093*.

Zhu, G. and Wei, P. (2019). Pre-departure planning for urban air mobility flights with dynamic airspace reservation. In *AIAA Aviation 2019 Forum*, page 3519.

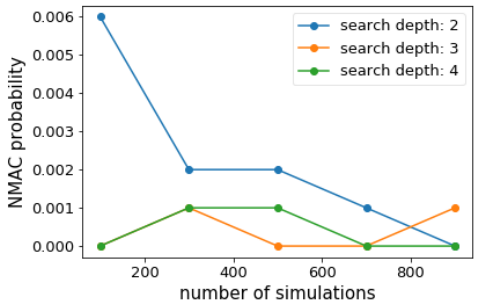
Zhu, P., Wen, L., Bian, X., Haibin, L., and Hu, Q. (2018). Vision meets drones: A challenge. *arXiv preprint arXiv:1804.07437*.

APPENDIX. PERFORMANCE OF MCTS ALGORITHM WITH DIFFERENT PARAMETERS

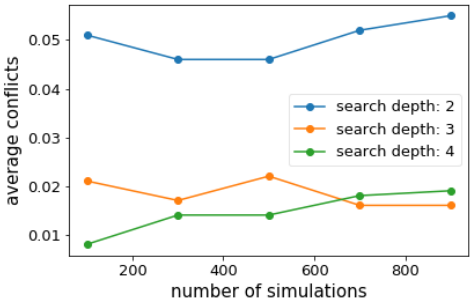
From Figure .2 to Figure .16, we list result for the MCTS algorithm, with the number of intruder aircraft ranging from 10 to 80, number of simulations ranging from 100 to 900, search depth ranging from 2 to 4. Each figure plots the result with a fixed number of intruder aircraft, and how the performance changes with different number of search depth and simulations.



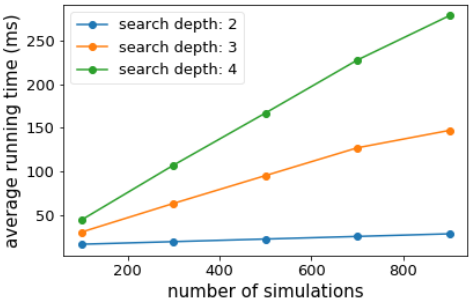
(a) Probability of reaching goal state



(b) Probability of having a NMAC

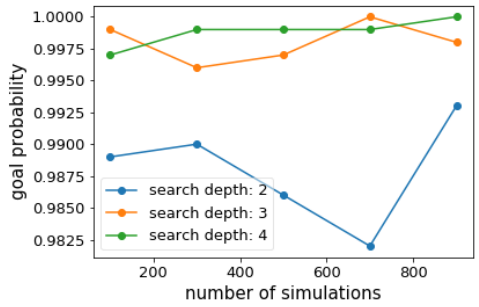


(c) Average conflicts in each episode

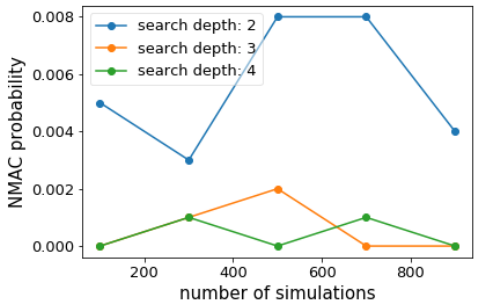


(d) Average running time for each decision

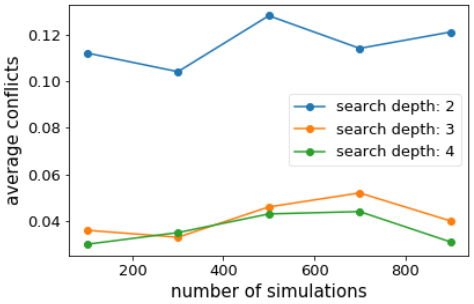
Figure .2: Performance of MCTS algorithm with different parameters when there is 10 intruder aircraft.



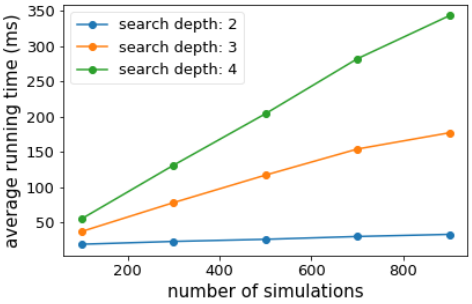
(a) Probability of reaching goal state



(b) Probability of having a NMAC

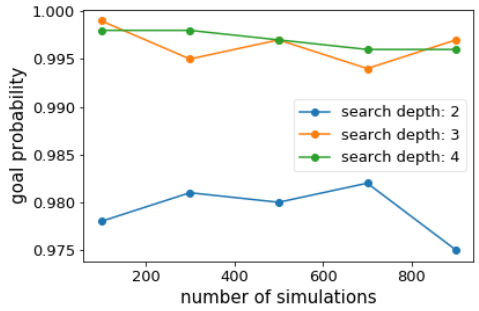


(c) Average conflicts in each episode

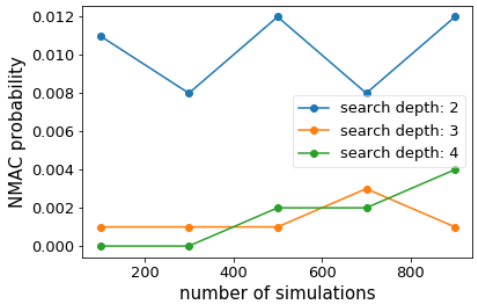


(d) Average running time for each decision

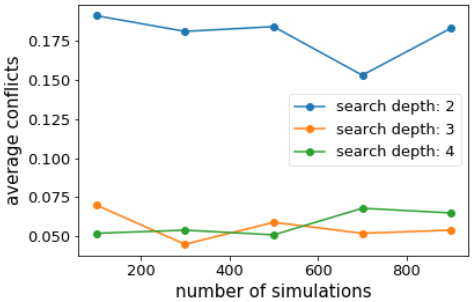
Figure .4: Performance of MCTS algorithm with different parameters when there is 20 intruder aircraft.



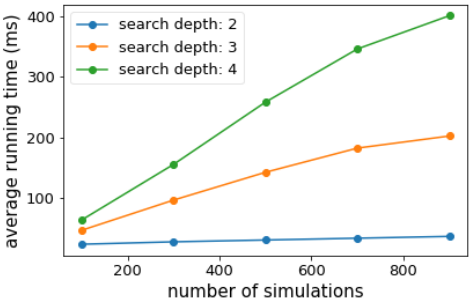
(a) Probability of reaching goal state



(b) Probability of having a NMAC

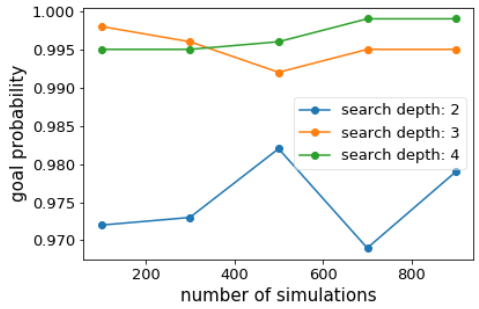


(c) Average conflicts in each episode

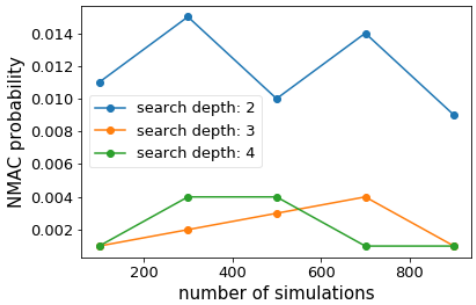


(d) Average running time for each decision

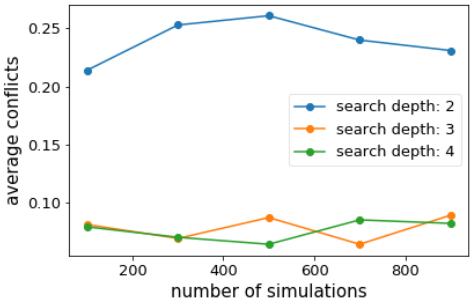
Figure .6: Performance of MCTS algorithm with different parameters when there is 30 intruder aircraft.



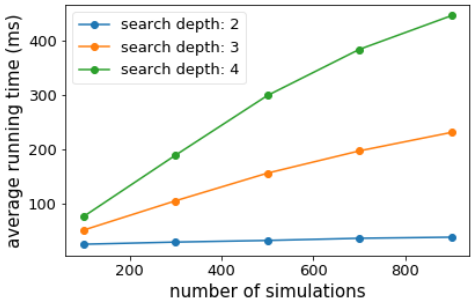
(a) Probability of reaching goal state



(b) Probability of having a NMAC

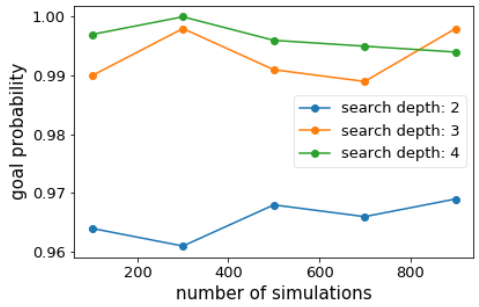


(c) Average conflicts in each episode

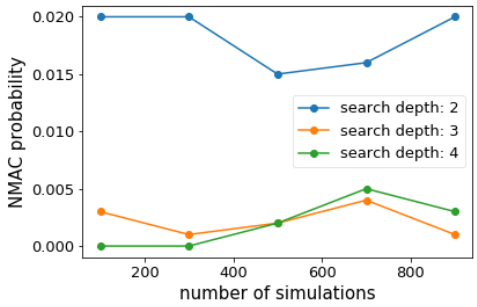


(d) Average running time for each decision

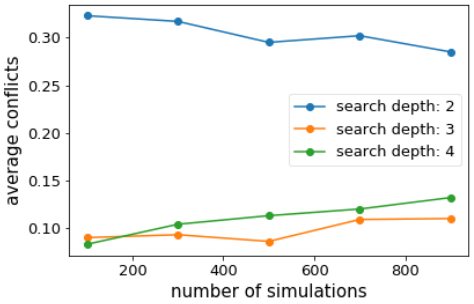
Figure .8: Performance of MCTS algorithm with different parameters when there is 40 intruder aircraft.



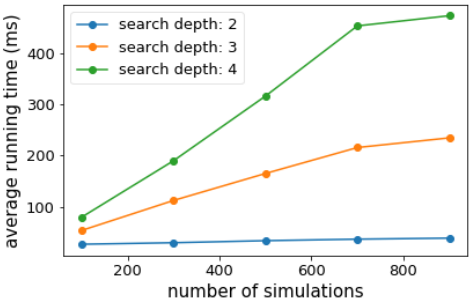
(a) Probability of reaching goal state



(b) Probability of having a NMAC

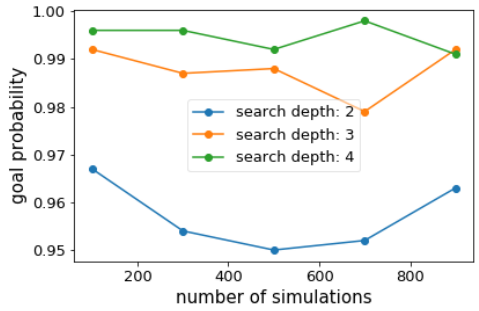


(c) Average conflicts in each episode

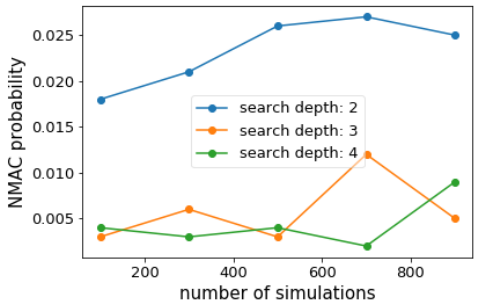


(d) Average running time for each decision

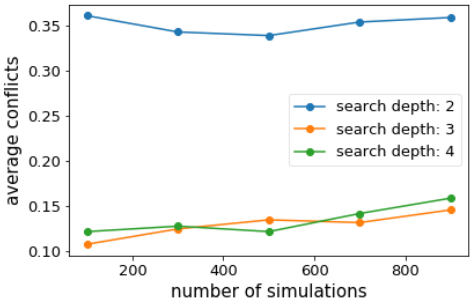
Figure .10: Performance of MCTS algorithm with different parameters when there is 50 intruder aircraft.



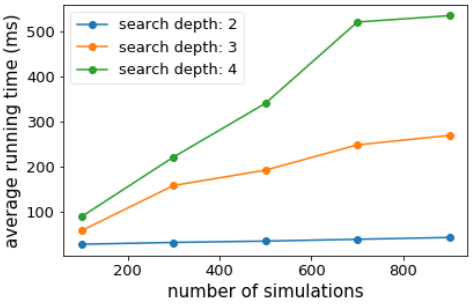
(a) Probability of reaching goal state



(b) Probability of having a NMAC

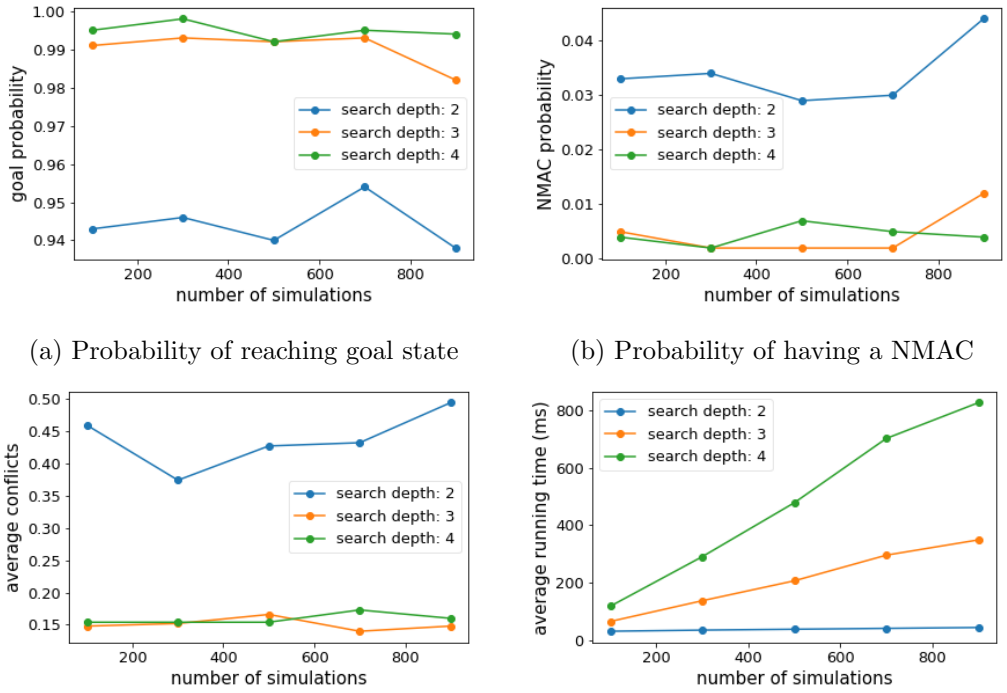


(c) Average conflicts in each episode



(d) Average running time for each decision

Figure .12: Performance of MCTS algorithm with different parameters when there is 60 intruder aircraft.



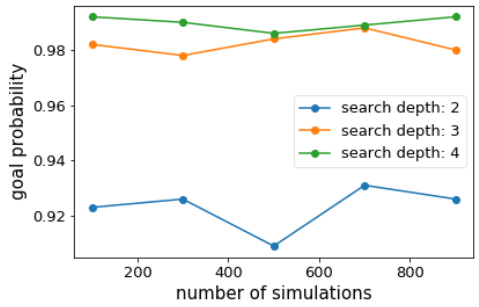
(a) Probability of reaching goal state

(b) Probability of having a NMAC

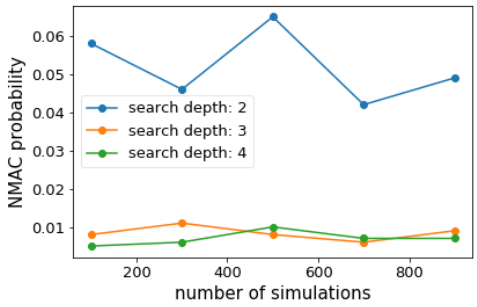
(c) Average conflicts in each episode

(d) Average running time for each decision

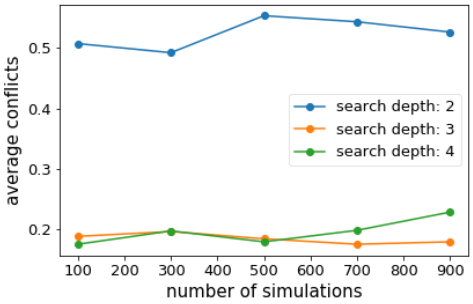
Figure .14: Performance of MCTS algorithm with different parameters when there is 70 intruder aircraft.



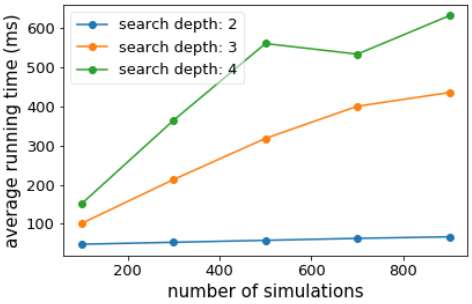
(a) Probability of reaching goal state



(b) Probability of having a NMAC



(c) Average conflicts in each episode



(d) Average running time for each decision

Figure .16: Performance of MCTS algorithm with different parameters when there is 80 intruder aircraft.